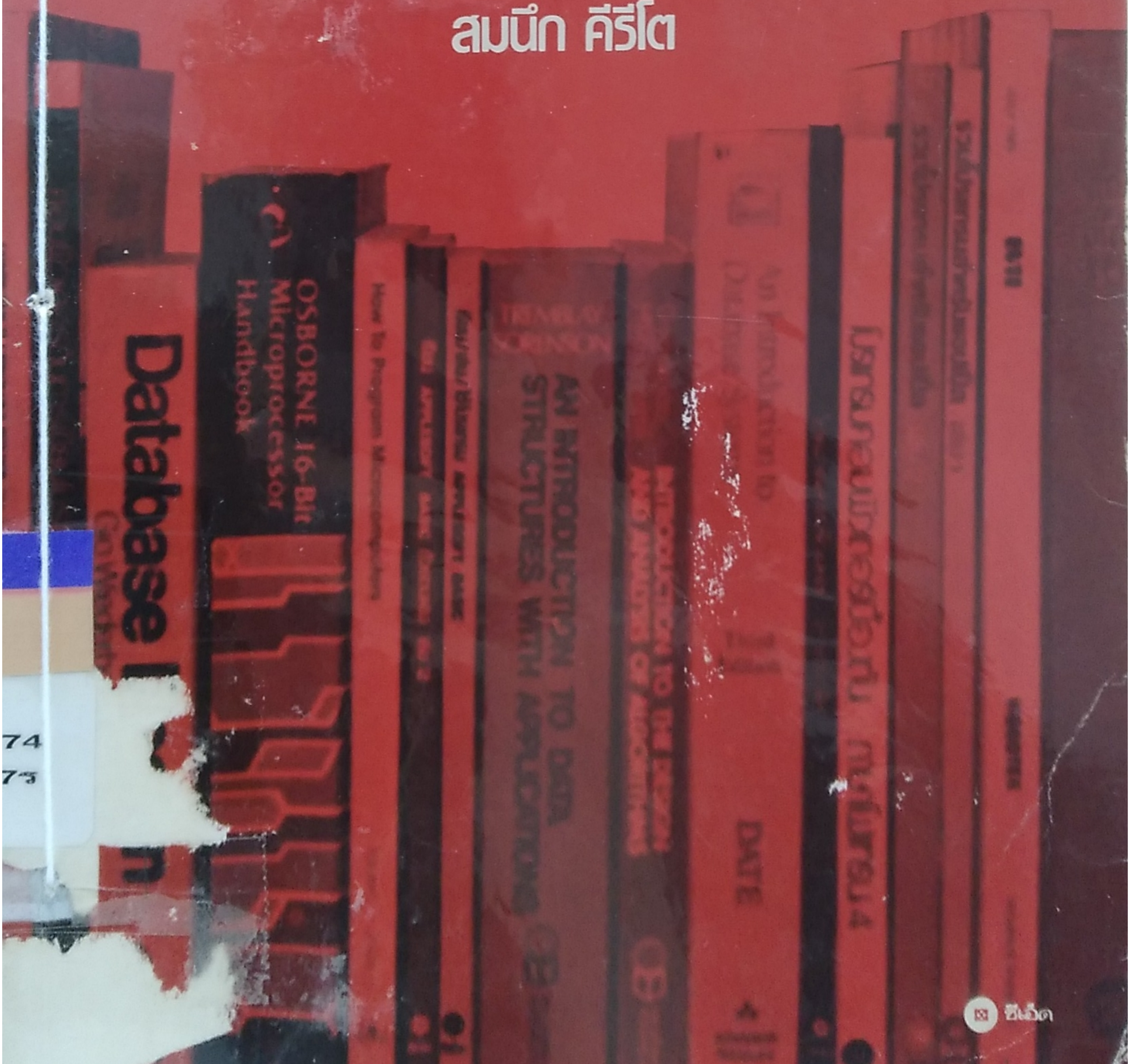
ระบบจัดการฐานข้อมูล dBASE II

ยืน ภู่วรวรรณ
สมนึก คีรีโต



ซีเอ็ด

ระบบจัดการฐานข้อมูล dBASE II

ยืน ภู่วรวรรณ
สมนึก คีรีโต

บริษัท ซีเอ็ดยูเคชั่น จำกัด
SCIENCE, ENGINEERING & EDUCATION CO., LTD.

ราคา 60 บาท

ISBN 974 – 7965 – 05 – 4

พิมพ์ครั้งที่ 1 พ.ศ. 2526

พิมพ์ครั้งที่ 2 พ.ศ. 2527

พิมพ์ครั้งที่ 3 พ.ศ. 2527

พิมพ์ครั้งที่ 4 พ.ศ. 2528

จัดพิมพ์และจัดจำหน่ายโดย

บริษัท ซีเอ็ดยูเคชั่น จำกัด

SCIENCE, ENGINEERING & EDUCATION CO.,LTD.

800/43-45 ซอยตระกูลสุข ถนนอโศกดินแดง

เขตห้วยขวาง กท. 10400 โทร. 245-0390

พิมพ์ที่ หจก. เอช-เอน การพิมพ์

402/16 ซอยดอนกุศล เขตยานนาวา กทม. 10120

นายประพันธ์ จิระมงคล ผู้พิมพ์ผู้โฆษณา พ.ศ. 2528

คำนำ

ข้อมูลเป็นปัจจัยสำคัญที่มีต่อระบบงานทุกวงการ ในบ้านก็เกี่ยวข้องกับข้อมูล ในที่ทำงานก็เกี่ยวข้องกับข้อมูล ในวงการธุรกิจเอกชน หรือแม้แต่หน่วยงานอุตสาหกรรมต่าง ๆ ก็เกี่ยวข้องกับข้อมูล สำหรับไมโครคอมพิวเตอร์นั้นเริ่มเข้ามามีบทบาทที่สำคัญต่อการจัดการข้อมูลมากขึ้น ปัจจุบันมีโปรแกรมสำเร็จรูปเป็นจำนวนมากที่สามารถใช้งานในการบริหารและจัดการข้อมูล เช่น PFS (Personal Filing System) การ์ดบ๊อกซ์ (Card box) Data Star, DBmaster, Condor และ dBASE II เป็นต้น แต่อย่างไรก็ตามการใช้งานของโปรแกรมสำเร็จรูปต่าง ๆ เหล่านั้นก็จำเป็นต้องมีพื้นฐานและหลักการทางการจัดการข้อมูลอยู่บ้างเพื่อจะใช้งานโปรแกรมสำเร็จรูปอย่างมีประสิทธิภาพ

ในบรรดาโปรแกรมสำเร็จรูปดังกล่าว dBASE II เป็นโปรแกรมที่พัฒนาโดยบริษัทแอชตันเทท ที่มีประสิทธิภาพสูงโปรแกรมหนึ่ง ข้อดีเด่นของโปรแกรมสำเร็จรูปนี้เห็นจะได้แก่ ระบบการจัดการข้อมูลที่มีประสิทธิภาพ เป็นภาษาที่เขียนได้ด้วยหลักการโปรแกรมโครงสร้างที่ง่ายต่อการเขียนและตรวจสอบ การสร้างความสัมพันธ์ระหว่างไฟล์ในรูปแบบระบบฐานข้อมูลทำได้ดี ด้วยเหตุนี้เองโปรแกรม dBASE II จึงเป็นโปรแกรมที่แพร่หลายและใช้งานกันอย่างมาก ในขณะเดียวกันก็มีผู้สร้างโปรแกรมสนับสนุนต่าง ๆ อีกมากเช่น dUTIL, QUICKCODE, dNAME, dSCAN, dGRAPH, dCODE และ dRUN เป็นต้น ระบบการจัดฐานข้อมูลด้วย dBASE II จึงมีแนวโน้มที่ค่อนข้างจะสดใสและคาดว่าจะแพร่หลายมากในหมู่ผู้ใช้ไมโครคอมพิวเตอร์

หนังสือระบบจัดการฐานข้อมูลด้วยไมโครคอมพิวเตอร์นี้มีจุดมุ่งหมายเพื่อสร้างฐานความเข้าใจเกี่ยวกับการสร้างระบบฐานข้อมูล โดยสามารถที่จะทำการวิเคราะห์ระบบงานและสร้างระบบงานด้วย dBASE II ได้ รายละเอียดของหนังสือจึงเป็นทั้งหลักการทางทฤษฎี และคู่มือ รวมทั้งตัวอย่างที่จะทำให้เกิดความเข้าใจในการเขียนโปรแกรมด้วยภาษา dBASE II

เนื้อหาที่สำคัญแบ่งแยกออกเป็น 3 ตอน คือ 3 บทแรก เป็นหลักการพื้นฐานเกี่ยวกับระบบจัดการฐานข้อมูลและหลักการเขียนโปรแกรมโครงสร้าง ส่วนบทที่ 4 ถึงบทที่ 12 ว่าด้วยหลักการสร้างระบบฐานข้อมูลโดยยึดหลักการสร้างระบบฐานข้อมูลของ dBASE II ส่วน 2 บทสุดท้ายคือบทที่ 13 และ บทที่ 14 เป็นตัวอย่างการประยุกต์ที่ได้ทดลองสร้างขึ้นเพื่อให้เข้าใจโดยบุคลากรของห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์มหาวิทยาลัยเกษตรศาสตร์

การศึกษาระบบจัดการข้อมูลด้วยไมโครคอมพิวเตอร์นี้อาจทำได้โดยที่ท่านไม่ต้องมีเครื่องคอมพิวเตอร์เลยก็ได้ แต่ถ้าท่านมีไมโครคอมพิวเตอร์ท่านสามารถเริ่มการทดลองดูและกระทำตามคำสั่งต่าง ๆ ตามที่กล่าวในหนังสือนี้ โดยสิ่งมุ่งหวังคือต้องการให้เกิดความเข้าใจเป็นลำดับแบบต่อเนื่อง ในที่นี้ได้ยกตัวอย่างไฟล์ที่ใช้และความสัมพันธ์ของไฟล์ ซึ่งท่านเองอาจจะสร้างไฟล์เหล่านั้นและกระทำตามได้

สำหรับระบบภาษาไทยนั้นทางห้องปฏิบัติการวิจัยไมโครได้พัฒนาภายใต้โปรแกรมจัดระบบงาน CP/M แล้ว จึงสามารถทำให้ท่านใช้งานในระบบฐานข้อมูลภาษาไทยได้อย่างมีประสิทธิภาพ และระบบจัดการเรื่องไฟล์ก็เหมือนกับที่กล่าวในหนังสือเล่มนี้ทั้งสิ้น

ผู้เขียนและทีมงานห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์หวังว่าการประยุกต์ใช้ไมโครคอมพิวเตอร์ในระบบจัดการฐานข้อมูลนี้จะเป็นการทำให้การใช้งานไมโครคอมพิวเตอร์มีประสิทธิภาพ และน่าใช้ขึ้นอีกมาก ซึ่งจะเป็นผลดีต่อส่วนรวมโดยทั่วกัน หากมีข้อบกพร่องหรือปัญหาใดกรุณาแจ้งมายังทางผู้เขียน ซึ่งจะได้ยึดถือเป็นแนวทางในการปรับปรุงต่อไปในอนาคต และก็หวังอีกด้วยว่าหนังสือระบบจัดการฐานข้อมูลด้วยไมโครนี้จะมีประโยชน์ต่อท่านบ้าง

ยืน ภู่วรวรรณ
สมนึก คีรีโต

สารบัญ

1

ระบบฐานข้อมูล ด้วยไมโครคอมพิวเตอร์

ชีวิตประจำวันของชาวไทยเราเกี่ยวข้องกับข้อมูลภาษาไทยทั้งสิ้น ในบ้านก็ต้องมีข้อมูลการใช้จ่าย การเก็บสิ่งของ ในร้านขายของหรือห้างสรรพสินค้าก็ต้องมีข้อมูลเกี่ยวกับการซื้อขาย การเก็บรักษา ในธุรกิจก็เต็มไปด้วยข้อมูลทั้งทางด้านบัญชีและการบริหาร หน่วยงานของรัฐตลอดไปจนถึงหน่วยงานบริหารของกรมกองต่าง ๆ ก็เต็มไปด้วยข้อมูล ลองนึกดูว่าถ้าข้อมูลทั้งหมดต้องเข้ามาเก็บในตัวคอมพิวเตอร์โดยให้คอมพิวเตอร์ช่วยประมวลผล ค้นหาแยกแยะและทำรายงานให้ ความสะดวกสบายจะเกิดขึ้นเพียงไร

ระบบฐานข้อมูลที่ดีเป็นระบบ ย่อมสร้างความเจริญให้แก่ประเทศชาติเป็นอย่างดี ในระบบของนักบริหารการเรียกหาข้อมูลที่ต้องการใช้เพื่อตัดสินใจต้องกระทำได้อย่างรวดเร็ว ในระบบธุรกิจก็ต้องการทำงานแข่งกับเวลา ต้องออกบิลออกหลักฐานให้ลูกค้าได้ทันทีโดยที่ไม่ต้องให้ลูกค้ารอนาน

ความสำคัญของ "ระบบฐานข้อมูล" จึงนับว่าจะเพิ่มบทบาทที่สำคัญขึ้นเรื่อย ๆ แต่อย่างไรก็ตามการเรียนรู้และการใช้ฐานข้อมูลก็ยังไม่แพร่หลาย การใช้คอมพิวเตอร์ยังไม่มีประสิทธิภาพ ประจวบกับโครงสร้างความยุ่งยากทางภาษาไทย ทำให้การใช้งานระบบฐานข้อมูลมีข้อยุ่งยากเพิ่มขึ้น ปัจจุบันไมโครคอมพิวเตอร์มีราคาถูกลง และขีดความสามารถเพิ่มขึ้น จึงมีผู้พยายามพัฒนาทำให้ใช้งานได้กว้างขวาง ระบบจัดการฐานข้อมูลจึงได้รับความสนใจ และนำมาประยุกต์กันมาก

ระบบข้อมูลโดยใช้ไมโครคอมพิวเตอร์กับจานแม่เหล็ก

เนื่องจากข้อจำกัดต่อการใช้งานในการเก็บข้อมูลของแผ่นจานแม่เหล็ก โดยแผ่นจานแม่เหล็กขนาด 5 นิ้ว จะเก็บข้อมูลในระบบการเก็บข้อมูลหน้าเดียวและมีความหนาแน่นธรรมดา (single side single density) ได้เพียงประมาณ 100,000 ตัวอักษร แต่อย่างไรก็ดีปัจจุบันความหนาแน่นของการเก็บข้อมูลเริ่มที่จะทำให้หนาแน่นยิ่งขึ้น ซึ่งสามารถเพิ่มความจุได้ถึงขนาด 600,000 ตัวอักษรในแผ่นจานแม่เหล็กขนาด 5 นิ้ว และ 2,400,000 ตัวอักษรในแผ่นจานแม่เหล็กขนาด 8 นิ้ว

ระบบฐานข้อมูลที่ดีจึงต้องหาทางเก็บข้อมูลให้มีประสิทธิภาพมีเนื้อที่ในการเก็บน้อย ค้นหาข้อมูลได้ง่าย สามารถเรียกข้อมูลออกมาได้อย่างถูกต้อง และปรับปรุงเปลี่ยนแปลงข้อมูลได้โดยง่ายโดยที่ข้อมูลตัวอื่นไม่สูญหายไปด้วย

โครงร่างของไฟล์

ในระบบความสัมพันธ์ของฐานข้อมูลที่กล่าวถึงนี้เป็นระบบฐานข้อมูลที่เรียกว่า *relational data base* ส่วนของข้อมูลจะเก็บไว้ในรูปไฟล์เสมือนตารางสองมิติ โดยความสัมพันธ์ของข้อมูลจะมีความสัมพันธ์ในเชิง *แถว (row)* และ *คอลัมน์ (column)*

พิจารณาตัวอย่างของไฟล์ ๆ หนึ่ง

	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3
	ID	ชื่อคนไข้	รหัสโรค
แถวที่ 1	230125	มะลิ บุญเกิด	7413
แถวที่ 2	230258	ดวงใจ นาเลิศ	6148

ในแต่ละแถวจะมีชื่อเรียกว่า *เรคอร์ด (record)*

ในแต่ละคอลัมน์มีชื่อ เรียกว่า *ฟิลด์ (field)*

ในแต่ละฟิลด์จะมีชื่อกำกับบอกไว้ ดังนั้นในส่วนของฐานข้อมูลจะมองเห็นเป็นตัวแปรที่จะนำมาใช้ต่อไป

สังเกตว่าการเกี่ยวข้องกับข้อมูลจะเกี่ยวข้องเป็นตัว ๆ โดยแต่ละเรคอร์ดจะมีความหมายในตัวเองโดยไม่เกี่ยวข้องกับลำดับเรคอร์ด

ลักษณะของข้อมูลในฐานข้อมูล

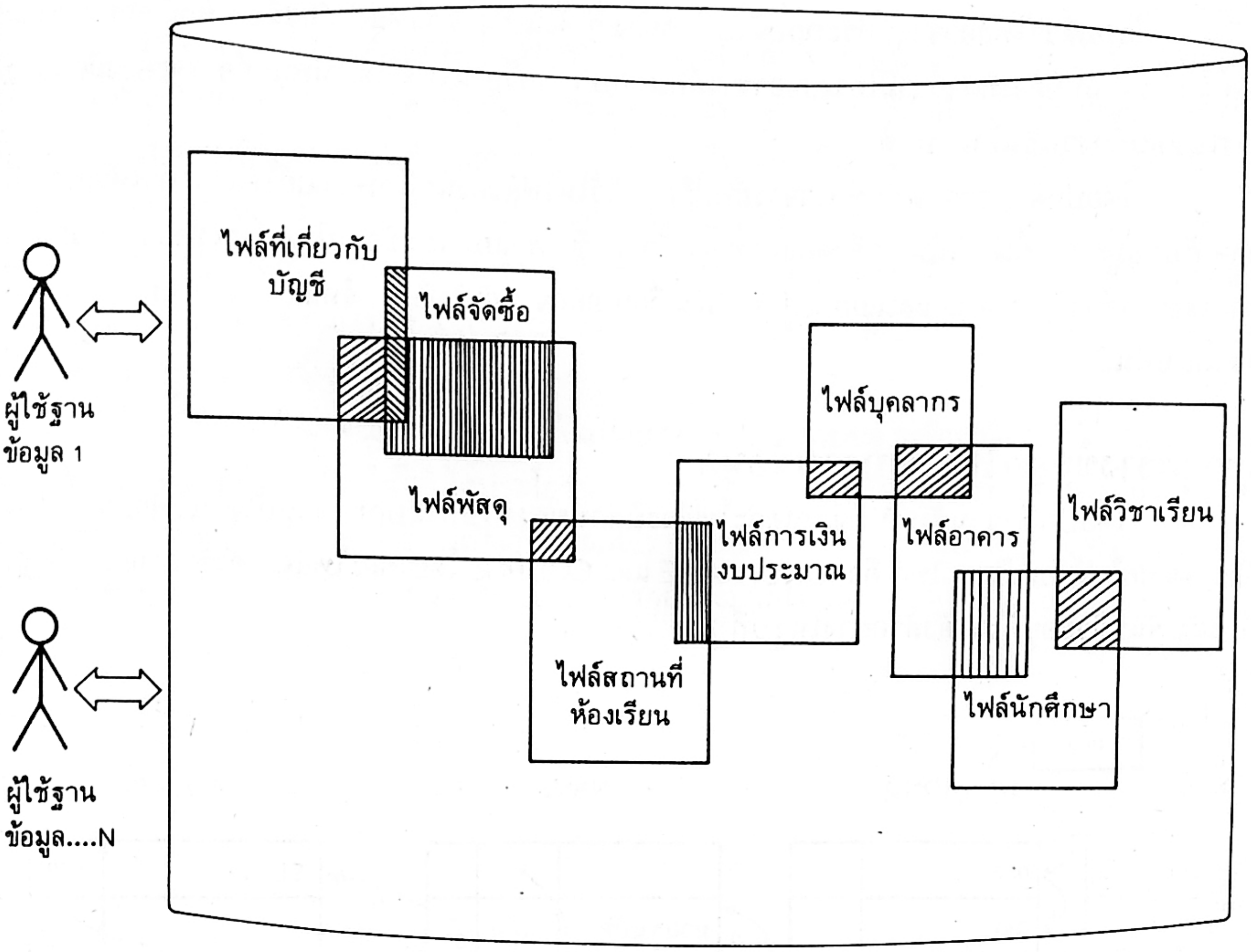
ระบบฐานข้อมูลจะประกอบด้วยไฟล์จำนวนหลาย ๆ ไฟล์ที่มีการจัดระบบข้อมูลในไฟล์ไว้ดีแล้ว การจัดระบบข้อมูลในไฟล์จะต้องไม่มีการซ้ำซ้อนของข้อมูลระหว่างไฟล์ต่าง ๆ และต้องเปิดโอกาสให้ผู้ใช้สามารถเข้าถึงข้อมูล ทำการค้นหาได้ง่าย เพิ่มเติม หรือลบออกก็ทำได้โดยถูกต้องไม่ทำให้ข้อมูลอื่นเสียหาย

ข้อมูลทั้งหมด คือ ข้อมูลที่ประกอบรวมอยู่ในฐานข้อมูล ส่วนของข้อมูลที่ซ้ำซ้อน คือส่วนของข้อมูลที่แรเงาไว้ การเก็บข้อมูลดังกล่าวอาจแยกอิสระเป็นไฟล์เลยก็ได้แต่มีส่วนชี้แสดงความสัมพันธ์ถึงกัน เช่น ระบบฐานข้อมูลของภาควิชาของสถาบันหนึ่งอาจเก็บได้เป็นดังรูปที่ 1.2

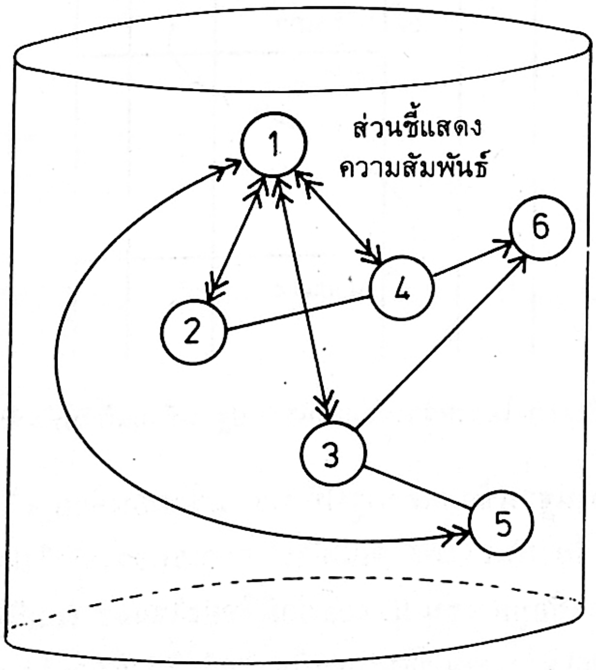
ไฟล์ข้อมูลที่ประกอบในฐานข้อมูล

ไฟล์ข้อมูลที่ประกอบในฐานข้อมูลได้แก่

1. ไฟล์อาจารย์
2. ไฟล์นักศึกษาปริญญาโท
3. ไฟล์นักศึกษาปริญญาตรี
4. ไฟล์โครงงานนักศึกษา
5. ไฟล์วิทยานิพนธ์
6. ไฟล์นักศึกษาเก่า



รูปที่ 1.1 เมื่อไฟล์หลาย ๆ ไฟล์ประกอบรวมกันเป็นฐานข้อมูล



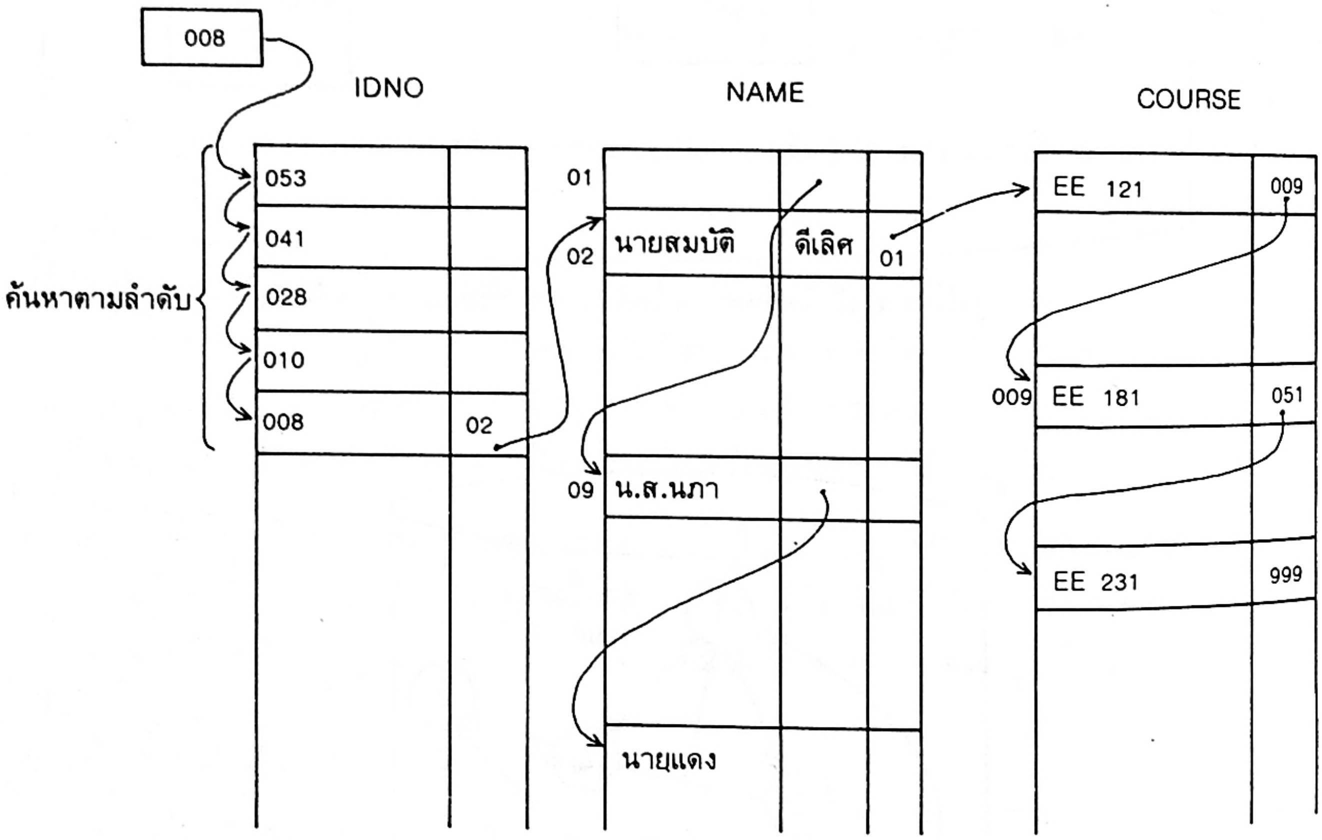
รูปที่ 1.2 แสดงความสัมพันธ์ของไฟล์แต่ละไฟล์ในฐานข้อมูล

สมมติว่าไฟล์อาจารย์ประกอบด้วยฟิลด์ต่าง ๆ ดังนี้ ชื่อ ตำแหน่ง เงินเดือน ที่อยู่ ฯลฯ ส่วนในไฟล์นักศึกษานั้นก็ต้องมีตัวชี้ว่ามีใครเป็นอาจารย์ที่ปรึกษา หรือในไฟล์วิทยานิพนธ์ก็อาจจะต้องมีตัวชี้ว่าใครเป็นคนทำงานชิ้นนั้น เป็นต้น

โดยปกติ เราอาจเก็บชื่ออาจารย์ที่ปรึกษาไว้ในไฟล์ของนักศึกษาเลยก็ได้แต่จะทำให้เสียเนื้อที่การเก็บข้อมูลมากขึ้น ดังนั้นเราจึงต้องหาทางสร้างตัวชี้ เพื่อแยกแยะข้อมูลในแต่ละเรคอร์ด ตัวอย่างเช่น เลขประจำตัว ชื่อ สกุล และแยกที่อยู่ อายุ เงินเดือน ออกจากกันไปอีกไฟล์หนึ่ง แต่จะมีคีย์คอยชี้บอกถึงความสัมพันธ์

โครงสร้างข้อมูลในระบบฐานข้อมูล

โดยปกติความสัมพันธ์ของแต่ละไฟล์จะมีส่วนของตัวชี้ที่จะบอกว่าอยู่ที่ใดในไฟล์อื่น ๆ เช่น เมื่อแบ่งแยกไฟล์ออกเป็น 3 ไฟล์ คือ IDNO, NAME และ COURSE โดยแต่ละไฟล์จะมีตัวชี้บ่งบอกว่าข้อมูลที่จะสัมพันธ์ด้วยอยู่ที่ใด ดังตัวอย่างในรูปที่ 1.3



รูปที่ 1.3 ตัวอย่างโครงสร้างข้อมูลในระบบฐานข้อมูลที่สัมพันธ์กันในแต่ละไฟล์

โครงสร้างข้อมูลในฐานข้อมูลตามรูปที่ 1.3 ประกอบด้วย 3 ไฟล์ ในแต่ละไฟล์ เช่น ในไฟล์ NAME ก็จะมีการจัดเรียงกันในไฟล์โดยมีตัวบอกว่าเรคอร์ดถัดไปอยู่ที่ไหน

กรณีที่ต้องการหาข้อมูล เช่น ID 008 ว่ามีชื่ออะไรและเรียนวิชาอะไร ลักษณะการค้นหาคือ จะใช้ค้นหาในไฟล์ IDNO ทีละเรคอร์ดจนพบก็จะได้ตัวชี้ว่าชื่ออะไร และโยงต่อว่าเรียนวิชาอะไร เป็นต้น การค้นหาข้อมูลจะเกิดขึ้นได้จำเป็นต้องมีการชี้ข้อมูลเพื่อให้เข้าถึงข้อมูลได้รวดเร็วขึ้น

ระบบฐานข้อมูลบนเครื่องไมโครคอมพิวเตอร์

ในระบบไมโครคอมพิวเตอร์มีขีดจำกัดทางด้านความจุของข้อมูล เพราะจานแม่เหล็กแบบอ่อนให้ความจุอยู่ในสเกลแสนตัวอักษรเท่านั้น การจัดวางระบบฐานข้อมูลที่ไม่ใหญ่โตนักย่อมเป็นไปได้

สำหรับข้อมูลของฐานข้อมูลส่วนใหญ่มักมีภาษาไทยเกี่ยวข้อง ทางห้องปฏิบัติการวิจัยได้ทำการพัฒนาฐานข้อมูลโดยสร้างฟังก์ชันการรับรู้ภาษาไทยในโปรแกรมสำเร็จรูป (DBMS-Data Base Management System) ซึ่งสามารถจัดระบบฐานข้อมูลภาษาไทยได้ การจัดลำดับการค้นหาหรือแม้แต่การจัดรูปแบบการพิมพ์สามารถทำได้โดยง่าย การออกแบบระบบฐานข้อมูลก็ทำได้โดยไม่จำเป็นต้องเรียนรู้ภาษา BASIC มาก่อนเลย

ระบบฐานข้อมูลจะเข้ามามีบทบาทในเกือบทุกวงการ ระบบฐานข้อมูลจึงถือได้ว่าเป็นส่วนของการที่จะนำเอาคอมพิวเตอร์มาใช้ให้เกิดประโยชน์คุ้มค่ามากที่สุด การพัฒนาการทางด้านไมโครคอมพิวเตอร์จะทำให้ระบบไมโครคอมพิวเตอร์เป็นอุปกรณ์ที่มีราคาไม่แพงนักแต่ทำงานได้เกินคุ้ม และเราท่านคงจะเห็นไมโครคอมพิวเตอร์เข้ามามีส่วนเกี่ยวข้องกับเราอย่างมากในอนาคตอันใกล้นี้

โครงสร้างข้อมูล

หากจินตนาการถึงการใช้ข้อมูลที่มักทำกันในปัจจุบัน โดยสมมติว่า โพลีคลีนิคแห่งหนึ่งต้องการเก็บรวบรวมข้อมูลคนไข้ที่เข้ามารับการรักษาในคลีนิคแห่งนี้ ข้อมูลที่ต้องการเก็บรักษาไว้ซึ่งได้แก่ประวัติส่วนตัวของคนไข้ อาการที่มารับการรักษา วิธีการรักษา และผลการรักษาหากเจ้าของคลีนิคต้องการเก็บข้อมูล ทางหนึ่งที่ทำกันก็คือการจดบันทึกข้อมูลทั้งหมดลงบนกระดาษและเก็บกระดาษนั้นไว้ ถ้ามีข้อความที่ซ้ำกัน เช่น ชื่อ และที่อยู่ของคนไข้ฯลฯ ผู้เขียนต้องเขียนทุกใบก็จะเป็นการเสียเวลา ดังนั้นเจ้าของคลีนิคมักไปจ้างโรงพิมพ์พิมพ์ฟอร์มกระดาษขึ้นมาเพื่อให้การกรอกข้อมูลง่ายขึ้น ลองพิจารณาตัวอย่างของฟอร์มที่คลีนิคแห่งหนึ่งใช้

กขค โพลีคลีนิค

ถนนงามวงศ์วาน บางเขน กทม. โทร. 1111222

หมายเลขคนไข้ ☐☐☐☐☐☐

ชื่อคนไข้..ที่อยู่..

อายุ................ เพศ.................. อาชีพ.. โทรศัพท์........................

ผู้อ้างถึงได้....................................น้ำหนัก........................ ส่วนสูง...................................

อาการที่มา...

การวินิจฉัย..

การรักษา...

ผลการรักษา...

....

เจ้าของคลีนิคลงทุนซื้อลิ้นชักใส่ฟอร์มกระดาษโดยสามารถจุได้เป็นจำนวนมาก ลักษณะของข้อมูลดังกล่าวได้รับการเก็บเรียงไว้ในลิ้นชักอย่างดี เมื่อมีคนไข้ใหม่เพิ่มขึ้นก็เพิ่มต่อเข้าได้อีก ลักษณะของข้อมูลดังกล่าวก็เหมือนกับไฟล์ที่ใช้ในทางคอมพิวเตอร์นี่เอง

คราวนี้หากมีใครนำบัตรคนไข้มาให้ท่านดู ข้อมูลที่อยู่ในบัตรนี้ก็แสดงความหมายต่าง ๆ ได้ การที่ข้อมูลจะแสดงความหมายได้จะต้องประกอบด้วยส่วนที่โรงพิมพ์พิมพ์บนกระดาษ กับส่วนที่เขียนข้อมูลลง ส่วนที่โรงพิมพ์พิมพ์ลงไปนี้คือส่วนที่จะอธิบายเนื้อหาข้อมูลให้ชัดเจน หรือใช้ประโยชน์ได้ ส่วนที่กรอกคือข้อมูล (data) การจะใช้งานได้ผลจึงต้องมีทั้งตัวข้อมูลและคำอธิบายเนื้อหาลักษณะข้อมูล

หากพิจารณาถึงการจัดการข้อมูล การจัดการข้อมูลจะรวมไปถึงการจัดเก็บข้อมูล การเรียกใช้ข้อมูล รวมถึงการวิเคราะห์ข้อมูลเพื่อผลการใช้งานบางอย่าง ลองพิจารณาการเรียกใช้ข้อมูล หากมีคนไข้คนหนึ่งมา และท่านต้องการค้นหาข้อมูลเดิมที่เคยมีของคนไข้คนนี้ ทางหนึ่งที่ทำได้คือหยิบดูบัตรข้อมูลนั้นทีละใบตั้งแต่ใบแรกจนพบ การค้นหาวิธีนี้อาจต้องเสียเวลาบ้าง แต่หากการจัดเก็บข้อมูลทำด้วยการจัดเรียงตามชื่อ เช่น เรียง ก ข ค....ไว้แล้ว เมื่อคนไข้มาก็ถามชื่อและค้นเรียงตามชื่อนั้นก็จะพบข้อมูลได้รวดเร็วขึ้น การจัดเก็บข้อมูลบางครั้งจึงต้องมีการนำเอาข้อมูลที่จะค้นหามาเรียงกันก่อน ข้อมูลส่วนที่จะค้นหาในกรณีนี้เรียกว่า *คีย์*

ในระบบฐานข้อมูลเป็นสิ่งที่จำเป็นและเกี่ยวข้องกับการใช้งานในชีวิตประจำวัน การตัดสินใจสำหรับนักบริหารจะกระทำได้รวดเร็วถ้ามีข้อมูลที่ถูกต้องและพร้อมเพรียงสนับสนุน ในประเทศที่เจริญแล้วคอมพิวเตอร์สามารถใช้เป็นเครื่องช่วยในการประมวลผลได้เป็นอย่างดี แต่อย่างไรก็ตามการประมวลผลด้วยคอมพิวเตอร์ก็จำเป็นต้องมีหลักการและวิธีการที่ผู้วางระบบงานเป็นผู้จัดไว้ หากไม่ได้กระทำให้เป็นระบบที่มีระเบียบแบบแผนที่ดีแล้วก็ยากที่จะใช้งานคอมพิวเตอร์ได้ผล

ข้อมูลคืออะไร

ข้อมูล คือ ความจริงที่เกี่ยวกับสิ่งต่าง ๆ เช่น คน สถานที่สิ่งของต่าง ๆ ซึ่งได้รับการรวบรวมเอาไว้ เมื่อข้อมูลได้รับการเก็บไว้จะสามารถเรียกมาใช้ประโยชน์ได้ในภายหลัง ข้อมูลจึงเป็นสิ่งที่ต้องมีการเก็บรวบรวมและรักษาไว้

ข้อสนเทศ (information) คือ สิ่งที่ได้จากการนำเอาข้อมูลที่เก็บรวบรวมไว้มาประมวลผล กล่าวคือข้อมูลที่ดีจักต้องมีการจัดเก็บข้อมูลไว้ได้เป็นอย่างดี มีการนำมาใช้ในการประมวลผล และควบคุมดูแลข้อมูลได้ง่าย การควบคุมดูแลอาจจะมีการกำหนดให้ผู้ใดบ้างเป็นผู้ใช้ข้อมูลได้ ข้อมูลที่เป็นความลับก็จะต้องมีระบบขั้นตอนที่ดี การแก้ไขหรือเกี่ยวข้องกับข้อมูลจะกระทำได้โดยใครบ้าง ซึ่งก็ต้องเป็นเรื่องที่ต้องมีการควบคุมเช่นกัน นอกจากนี้ข้อมูลที่เก็บไว้แล้วต้องไม่เกิดการสูญหายหรือถูกทำลายลงไปได้ โดยคุณสมบัติของข้อมูลที่ดีจะต้องไม่มีส่วนการเก็บข้อมูลซ้ำซ้อน เพราะจะเป็นการสิ้นเปลืองเนื้อที่เก็บข้อมูล มีรูปลักษณะง่ายต่อการเก็บมีความเป็นรูปแบบเดียวกัน ข้อมูลแต่ละชุดมีความหมายในตัวเองหรือมีความอิสระในตัวเอง

การเก็บข้อมูลนั้นผู้ทำการเก็บจำเป็นต้องทำการแยกแยะ และพยายามหาทางลดขนาดของข้อมูลให้สั้นที่สุดแต่ได้ความหมายในตัวเองให้มากที่สุด และโดยปกติข้อมูลที่ต้องการเก็บมีเป็นจำนวนมาก เช่น บริษัทแห่งหนึ่งต้องการเก็บข้อมูลอาจจะมีข้อมูลให้เก็บได้เป็นจำนวนหลายไฟล์ การเก็บข้อมูลจึงจำเป็นต้องแยกกรุ๊ฟ หรือกลุ่มออกจากกัน แต่ข้อมูลในระหว่างกลุ่มก็อาจจะมีส่วนประสานสัมพันธ์หรือเกี่ยวข้องกันได้ ความสัมพันธ์ระหว่างกลุ่มนี้เองเป็นส่วนที่ทำให้เกิดระบบฐานข้อมูล (database) ซึ่งเป็นศาสตร์ที่จำเป็น

ต้องเข้าใจหลักการและวิธีการเพื่อให้เกิดการเก็บ เรียกหา ค้นหา หรือใช้งานข้อมูลได้ประสิทธิภาพสูงสุด เมื่อเป็นเช่นนี้ทางคอมพิวเตอร์จึงกำหนดการแยกกลุ่มข้อมูลออก โดยยึดหลักการพื้นฐานที่ข้อมูลแต่ละกลุ่มจะเป็นสิ่งที่มองเห็นหรือจับต้องได้ เช่น คน สิ่งของ สินค้า สถานที่ เป็นต้น การแยกแยะกลุ่มนี้ในแต่ละกลุ่มเราเรียกว่า *เอ็นทิที (entity)* โดยสรุปเอนทิทีหมายถึงสิ่งที่เราสามารถมองเห็นหรือสร้างภาพพจน์ได้ โดยข้อสนเทศของเอ็นทิทีจะสามารถแยกออกได้เป็นสองส่วน คือ เนื้อหาและข้อมูล

สำหรับเนื้อหาของเอ็นทิทีชนิดเดียวกันจะมีลักษณะเหมือนกัน แต่ส่วนของข้อมูลแตกต่างกันออกไป เนื้อหาจึงเป็นส่วนที่จะบอกรายละเอียดเพื่อขยายข้อมูลให้ได้ความหมายครบถ้วนยิ่งขึ้น พิจารณาจากระบบข้อมูลโดยดูตัวอย่างเอ็นทิทีของบุคลากรของบริษัทแห่งหนึ่ง

เอ็นทิที	เนื้อหา	ข้อมูล
คน	ชื่อคน	นายแดง
	อาชีพ	คนขับรถ
	ที่ทำงาน	บริษัท เจริญกิจ
	อายุ	35 ปี

โดยปกติเอ็นทิทีต่างชนิดกันก็จะมีเนื้อหาที่แตกต่างกันออกไปด้วย ตัวอย่างเช่น เอ็นทิทีของสินค้าก็จะมีรายละเอียดดังนี้

เอ็นทิที	เนื้อหา	ข้อมูล
สินค้า	รหัสสินค้า	0253
	ชื่อสินค้า	ทรานซิสเตอร์
	จำนวนสินค้า	100 ตัว
	ราคาสินค้า	15 บาท

ระบบฐานข้อมูลกับคอมพิวเตอร์

ในการนำเอาคอมพิวเตอร์มาจัดการระบบฐานข้อมูลนั้น ข้อมูลของเอ็นทิทีต่าง ๆ จะได้รับการนำเอาไปเก็บไว้ในหน่วยความจำของคอมพิวเตอร์ ดังนั้นข้อมูลที่เก็บไว้อาจได้รับการเปลี่ยนแปลงเพิ่มเติม หรือตัดออกได้ และการเก็บข้อมูลนั้นจะทำการเก็บข้อมูลไว้ได้หลาย ๆ เอ็นทิที แต่เมื่อเรียกใช้อาจนำเอาข้อมูลจากหลาย ๆ เอ็นทิทีนั้นมาสัมพันธ์กันเพื่อให้ผลลัพธ์ที่ต้องการนั่นเอง

ดังนั้นในการเก็บข้อมูลเข้าในส่วนหน่วยความจำของคอมพิวเตอร์นั้น จำเป็นต้องบ่งบอกลักษณะของข้อมูลชนิดต่าง ๆ ของเอ็นทิทีนั้น ๆ ให้แน่นอน โดยปกติการกำหนดลักษณะของข้อมูลนั้นจะกำหนดในรูปแบบของการเป็นตัวอักษรหรือตัวเลข โดยพิจารณาจากตัวอย่างเอ็นทิทีลูกค้า

การมองลักษณะของเอ็นทิทีดังได้กล่าวนี้อาจมองในรูปแบบของไฟล์ดังได้กล่าวถึงแล้วในบทที่ 1 ก็ได้ กล่าวคือไฟล์ลูกค้า ซึ่งรายละเอียดของข้อสนเทศที่จะนำมาใช้ได้ต้องประกอบด้วยเนื้อหา ข้อ

เนื้อหา	ข้อมูล	ลักษณะของข้อมูล
รหัสลูกค้า	832501	ตัวเลข 6 ตัว
ชื่อลูกค้า	บริษัทร่วมค้า จำกัด	อักษร 30 ตัว
ที่อยู่	235/8 ถนนเพชรบุรี	อักษร 30 ตัว
โทรศัพท์	2253581	อักษร 7 ตัว
รหัสสถานะ	4	ตัวเลข 2 ตัว

มูล และลักษณะของข้อมูล สำหรับลักษณะของข้อมูลในไฟล์เราเรียกว่า *โครงสร้างของไฟล์ (file structure)* ส่วนตัวข้อมูลที่เก็บนี้จะเป็นข้อมูลที่เก็บไว้ในหน่วยความจำนั่นเอง

ในการใช้งานด้วยระบบคอมพิวเตอร์นั้น ข้อมูลจะได้รับการเก็บไว้เพื่อให้ได้ประสิทธิภาพคือกินเนื้อที่น้อย และจะต้องเรียกค้นหาข้อมูลได้โดยง่าย ดังนั้นจึงมีการแบ่งแยกข้อมูลออกเป็นส่วน ๆ ตามเนื้อหาของเอ็นทิที ส่วนย่อยของเอ็นทิทีนี้เรียกว่า *ฟิลด์ (field)* โดยมีการเรียกใช้ข้อมูล

พิจารณาโครงร่างเอ็นทิทีลูกค้า

เนื้อหา		ข้อมูล	ลักษณะของข้อมูล	
ความหมาย	ชื่อฟิลด์		ชนิด	ความกว้าง (ตัวอักษร)
รหัสลูกค้า	IDNO	832501	N	6
ชื่อลูกค้า	NAME	บริษัทร่วมค้า	C	30
ที่อยู่	ADDR	235/8 ถนนเพชรบุรี	C	30
โทรศัพท์	TELNO	2253581	C	7
รหัสสถานะ	STATUS	4	N	2

เมื่อนำเอาฟิลด์ของเอ็นทิทีทุก ๆ ฟิลด์มาวางเรียงกันจะเกิดรูปแบบที่ทางคอมพิวเตอร์มองเห็นที่เรียกว่า *เรคอร์ด (record)* ซึ่งใช้เป็นเครื่องบ่งบอกถึงเอ็นทิทีนั้น ๆ ได้ เช่น เอ็นทิทีลูกค้า มีโครงสร้างเรคอร์ดเป็น

IDNO	NAME	ADDR	TELNO	STATUS

ในเรคอร์ดแต่ละอันจะมีฟิลด์อยู่หนึ่งอันจะบ่งบอกความแตกต่างของข้อมูลให้เราทราบได้เราจะเรียกฟิลด์อันนี้ว่า *คีย์ (key)* เช่น เรคอร์ดของลูกค้าจะมี *NAME เป็นคีย์* เพราะชื่อนี้จะบอกแยกแยะความแตกต่าง กล่าวคือจะบอกได้ว่าเป็นลูกค้ารายเดียวกันหรือไม่ ถ้าข้อมูลใน NAME เหมือนกัน แสดงว่าเป็นรายเดียวกัน แต่ถ้าหากค่าต่างกันจะแสดงว่าเป็นคนละรายกัน

โครงสร้างของข้อมูล

ในเรคอร์ดของเอ็นทิทีแต่ละชนิดนั้นจะแสดงถึงโครงสร้างข้อมูล โดยมีลักษณะการเก็บข้อมูลที่บ่งบอกจำนวนตัวอักษร โดยปกติในแต่ละฟิลด์ที่ประกอบกันขึ้นนั้นจะมีความสัมพันธ์ซึ่งกันและกัน ความ

สัมพันธ์นี้จะก่อให้เกิดเป็นโครงสร้างข้อมูลที่เป็นแบบต่าง ๆ กันขึ้น โดยโครงสร้างเหล่านี้แยกออกได้เป็น 3 แบบ ด้วยกัน

โครงสร้างแบบธรรมดา เป็นโครงสร้างที่พื้นฐานที่สุด โดยฟิลด์แต่ละอันในเรคอร์ดจะขึ้นอยู่กับฟิลด์ที่เป็นคีย์เท่านั้น กล่าวคือ ข้อมูลของฟิลด์อื่น ๆ จะมีความหมายใช้การได้ก็ต่อเมื่อเรารู้ข้อมูลที่เป็นคีย์แล้วเท่านั้น ตัวอย่างเช่น ข้อมูลของเอ็นทิทีพนักงาน โดยมีรายละเอียดโครงสร้างฟิลด์ เป็น

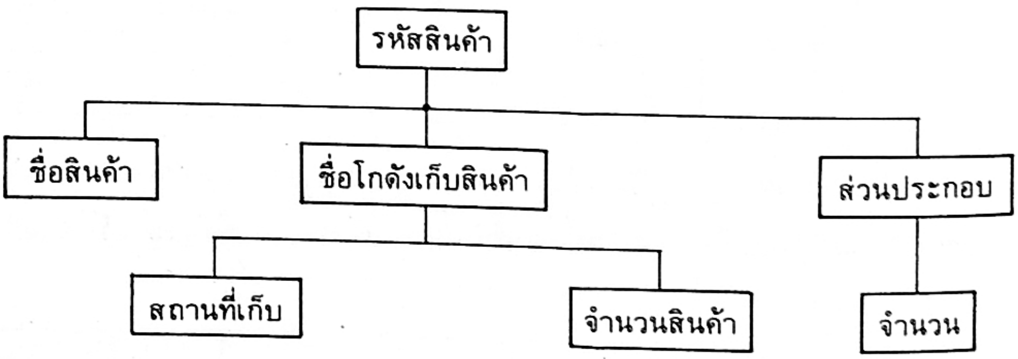
IDNO	ชื่อ	ที่อยู่	วันเข้ารับทำงาน	ประจำแผนก	เงินเดือน

ตัวอย่างข้อมูล เช่น

81253	นายมี	231/5 ถนนงามวงศ์วาน	3/05/22	ผลิต	2500
82258	นายมา	21 ถนนพหลโยธิน	5/12/18	ขาย	3100
83256	นายมี	5/1 ถนนเพชรบุรี	3/05/22	ผลิต	2500

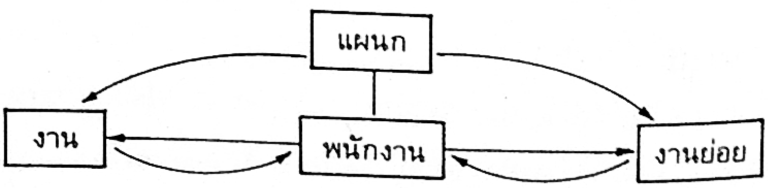
จากกรณีนี้เห็นได้ว่า ถ้าจะกำหนดคีย์ในที่นี้ IDNO จะเป็นคีย์ กล่าวคือถ้าหาก IDNO ต่างกันจะแสดงความหมายถึงการเป็นคนละคนได้ แต่ถ้าจะดูเฉพาะชื่อจะเห็นว่ามีนายมีสองคน อยู่แผนกเดียวกันเงินเดือนเท่ากัน รับเข้าทำงานพร้อมกัน ข้อมูลในฟิลด์ต่าง ๆ จะบอกไม่ได้เลยจนกว่าจะทราบคีย์

โครงสร้างข้อมูลแบบลำดับขั้น เป็นโครงสร้างที่ใช้กันมากในโครงสร้างภาษาโคบอลลักษณะเช่นนี้คือฟิลด์บางอันในเรคอร์ดจะขึ้นกับฟิลด์อันอื่นที่ไม่ใช่คีย์อีกด้วย ตัวอย่างเช่น เอ็นทิทีของสินค้าที่อยู่ในโรงเก็บ

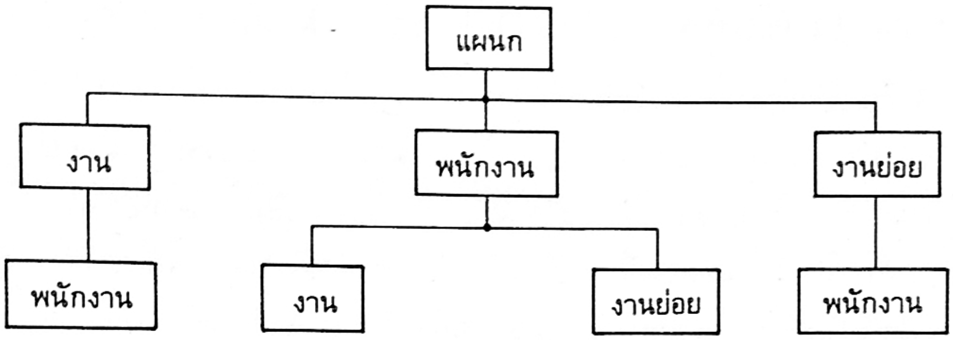


จากตัวอย่างนี้คงจะเห็นได้ว่าฟิลด์จำนวนสินค้าจะมีความหมายได้ก็ต่อเมื่อรู้ข้อมูลของฟิลด์เลขรหัสสินค้า และฟิลด์ชื่อโกดังเก็บสินค้าเสียก่อน การเกี่ยวข้องของข้อมูลจะเป็นลำดับขั้นเรียงกันขึ้นไป

โครงสร้างข้อมูลแบบเนทเวอร์ค โครงสร้างแบบนี้จะคล้ายกับแบบลำดับขั้น คือฟิลด์บางอันจะขึ้นกับฟิลด์อันอื่นที่ไม่ใช่คีย์ด้วยแต่จะแตกต่างกับโครงสร้างแบบลำดับขั้น หรือไม่เป็นรูปร่างของต้นไม้ (tree) แต่จะเกิดเป็นวงรอบ (loop) แทนพิจารณาโครงสร้างเอ็นทิทีของหน่วย (แผนก) ต่อไปนี้ดู



โครงสร้างแบบเนทเวอร์คนี้เป็นโครงสร้างที่ซับซ้อนมาก แต่ก็สามารถทำให้ง่ายขึ้นโดยเปลี่ยนโครงสร้างเป็นแบบลำดับขั้นได้



จะเห็นได้ว่าแผนกเป็นคีย์ของเรคอร์ดซึ่งข้อมูลในฟิลด์อื่นต้องขึ้นด้วย แต่งานและงานย่อยก็ต้องขึ้นกับพนักงานเสียก่อน

การกำหนดคีย์จึงเป็นเรื่องที่จะต้องพิจารณาเป็นอันดับแรกของโครงสร้างเอ็นทิที และในการสร้างโครงสร้างข้อมูลของระบบฐานข้อมูลที่จะกล่าวถึงต่อไปนี้จะต้องแยกแยะเอ็นทิทีออกให้ได้เสียก่อน แล้วนำเอ็นทิทีมาสร้างไฟล์โดยกำหนดคีย์และฟิลด์ต่าง ๆ ขึ้นมา โดยกำหนดโครงสร้างของไฟล์ตามรูปแบบที่ต้องการใช้เก็บข้อมูล

การแบ่งแยกไฟล์ตามลักษณะการเรียกหาข้อมูล

ในการเก็บข้อมูลด้วยคอมพิวเตอร์ในรูปแบบไฟล์นั้นคงพอนึกออกได้ว่าต้องประกอบด้วยลักษณะที่แทนด้วยชื่อฟิลด์รวมกันเป็นเรคอร์ด การเก็บข้อมูลจะเก็บทีละเรคอร์ดในการเรียกข้อมูลก็เช่นเดียวกันจะมีการอ่านข้อมูลทีละเรคอร์ดเช่นเดียวกัน การแบ่งแยกไฟล์จึงพอแบ่งแยกตามรูปแบบลักษณะการเรียกค้นหา หรือเขียนอ่านข้อมูลทีละเรคอร์ดบนที่เก็บข้อมูลของไฟล์นั่นเอง การแบ่งแยกจึงแบ่งออกเป็น 3 แบบด้วยกันคือ

ไฟล์ที่มีการเขียนอ่านข้อมูลเป็นลำดับ (sequential file) เป็นโครงสร้างการเก็บข้อมูลแบบพื้นฐานที่สุด กล่าวคือเมื่อมีการเขียนเพิ่มลงในหน่วยความจำทีละเรคอร์ด ข้อมูลจะเข้าต่อท้ายเรียงกันไป ในการอ่านข้อมูลก็จะอ่านเรียงกันไปทีละเรคอร์ดจากเริ่มแรกเป็นต้นไป การใช้งานค้นหาข้อมูลจึงต้องเรียงกันตั้งแต่เรคอร์ดแรกเป็นต้นไปจึงทำให้เชื่องช้า แต่ข้อดีของไฟล์ชนิดนี้คือความง่ายในการใช้งาน

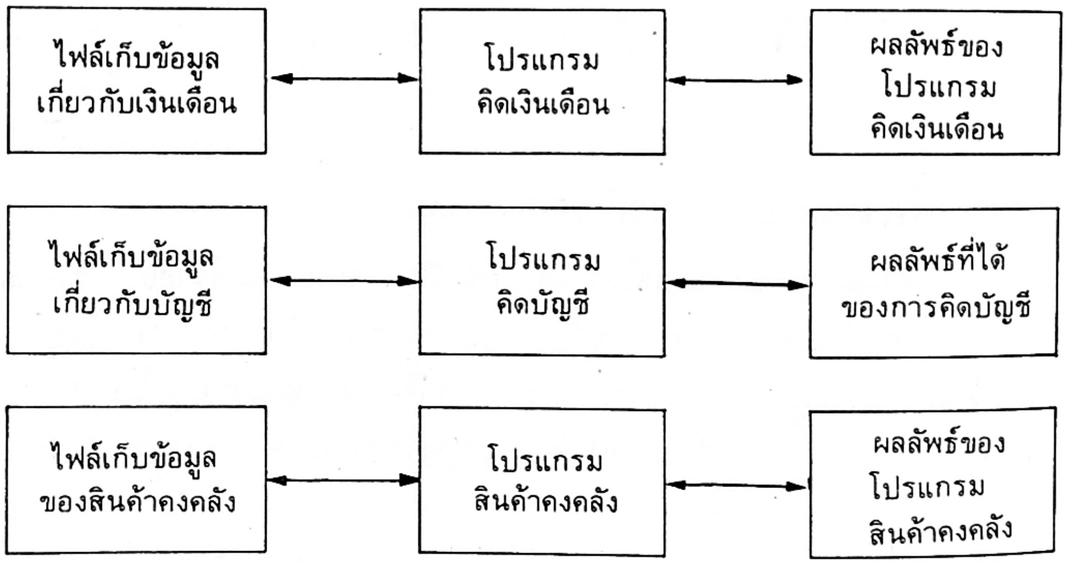
ไฟล์แบบแรนดอมแอคเซส (ramdom access file) เป็นไฟล์ที่มีคุณสมบัติซึ่งผู้ใช้สามารถอ่านหรือเขียนข้อมูลในตำแหน่งใด ๆ ของไฟล์ได้อย่างรวดเร็วโดยไม่ต้องเรียงลำดับจากต้นไฟล์ แต่อย่างไรก็ตามไม่สามารถเรียกตามคีย์ของเรคอร์ดนั้นได้โดยตรง ถ้าจะเรียกได้ต้องรู้หมายเลขเรคอร์ดเสียก่อน

ไฟล์แบบอินเด็กซ์ (index file) ไฟล์แบบนี้จำเป็นต้องมีการจัดเรียงข้อมูลของฟิลด์คีย์อย่างมีระเบียบเสียก่อนเพื่อประโยชน์การค้นหา การเรียก การเขียน การอ่าน เรคอร์ดใด ๆ สามารถใช้คีย์ฟิลด์เป็น

ตัวกำหนดได้โดยตรงทำให้สะดวกต่อการใช้เป็นอย่างมาก เป็นโครงสร้างไฟล์ที่มีในระบบจัดการฐานข้อมูล แต่ไม่มีในระดับตัวแปลภาษาเบสิค (ในไมโครคอมพิวเตอร์)

ระบบจัดการฐานข้อมูล (DBMS-Data Base Management System)

อย่างไรก็ตาม แม้เราจะมีเครื่องคอมพิวเตอร์ที่มีประสิทธิภาพดีแล้วก็ตาม เรายังต้องอาศัย "โปรแกรม" ที่จะควบคุมการทำงานของเครื่องอีกด้วย สำหรับบุคคลที่ไม่ค่อยได้คุ้นเคยในการเขียนโปรแกรมไม่ว่าจะใช้ภาษาคอมพิวเตอร์อะไรก็ตาม เช่น ภาษาเบสิค ภาษาฟอร์แทรน อาจจะต้องประสบกับปัญหาต่อการเขียนโปรแกรมที่ต้องเกี่ยวกับไฟล์ของข้อมูลชุดคำสั่งที่ยุ่งยากสลับซับซ้อน ตัวอย่างเช่น ถ้าเราอยู่ในโรงงานเห่งหนึ่งซึ่งต้องการจะเก็บรวบรวมข้อมูลเกี่ยวกับเงินเดือนของพนักงาน ข้อมูลเกี่ยวกับการทำบัญชีรายรับ รายจ่าย ข้อมูลของระบบสินค้าคงคลัง ในระบบการทำงานโดยทั่ว ๆ ไป เมื่อเราเข้าไปเกี่ยวข้องในการเขียนโปรแกรม หรือใช้งานคอมพิวเตอร์เพื่อให้ได้จุดประสงค์ ตามความต้องการดังกล่าว เราอาจจะใช้หลักการทำงานโดยวิธีการจัดไฟล์โดยทั่ว ๆ ไป ซึ่งเราเรียกวิธีนี้โดยทั่ว ๆ ไปว่า *file handing system* ดังรูปที่ 2.1



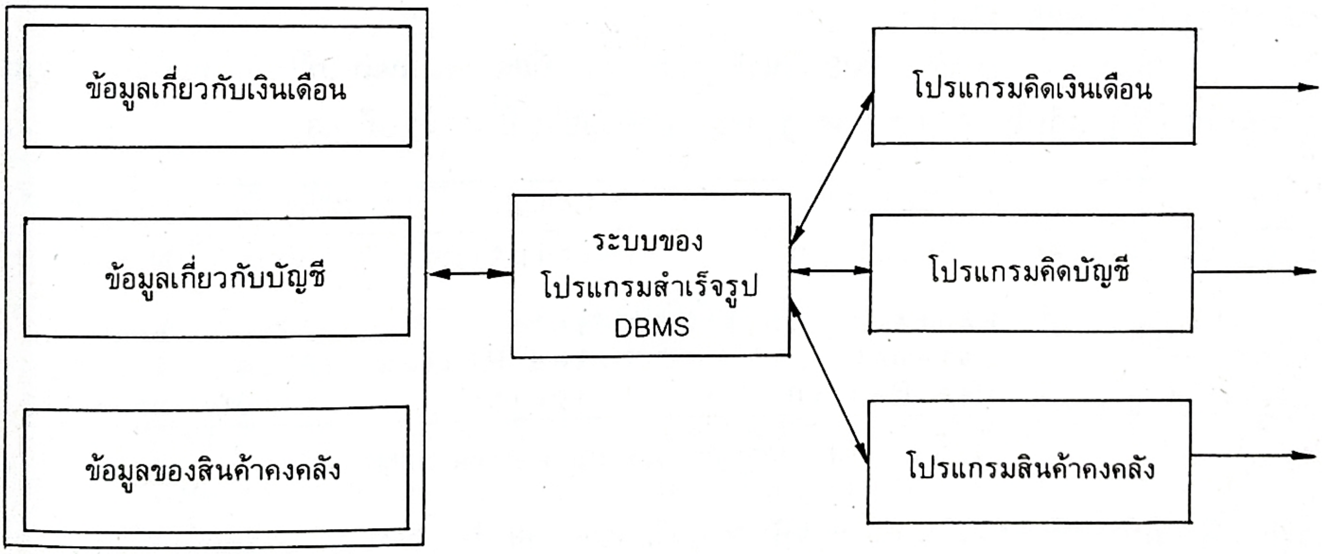
รูปที่ 2.1 ลักษณะหลักการของขบวนการจัดการเก็บไฟล์โดยทั่วไป

เราจะต้องเขียนโปรแกรมที่จะคำนวณเกี่ยวกับเงินเดือน ซึ่งจะทำการดึงเอาข้อมูลมาจากไฟล์ซึ่งเก็บข้อมูลเกี่ยวกับเงินเดือนมาทำการประมวลผล ส่วนโปรแกรมซึ่งทำการคำนวณทางด้านบัญชีก็จะต้องติดต่อกับไฟล์ที่เก็บข้อมูลของบัญชีและโปรแกรมที่ควบคุมสินค้าคงคลังก็จะเกี่ยวข้องกับไฟล์ของข้อมูลสินค้าคงคลัง เมื่อเราต้องการโปรแกรมเพื่อจุดประสงค์อะไรก็แล้วแต่เราก็ต้องเขียนโปรแกรมนั้นขึ้นมาเอง พร้อมกับต้องสร้างไฟล์ ซึ่งจะต้องเก็บข้อมูลที่เราต้องการนำมาประมวลผลอีกด้วย

เมื่อมาถึงจุดนี้ คงจะเห็นว่ากว่าจะได้ใช้เครื่องคอมพิวเตอร์ให้มีประสิทธิภาพจริง ๆ คงต้องจ้างผู้มีความรู้พิเศษทางด้านคอมพิวเตอร์มาทำงานด้านนี้โดยเฉพาะ...แต่อย่าเพิ่งตกใจปัจจุบันนี้ในระบบไมโครคอมพิวเตอร์เรามีเครื่องทุ่นแรงที่เรียกว่า *"โปรแกรมสำเร็จรูป"* ซึ่งจะช่วยผู้ใช้งานทางด้านนี้โดย

เฉพาะ ผู้ใช้อาจจะใช้เวลาในการเรียนรู้วิธีใช้งานอีกไม่นานนัก ก็สามารถที่จะใช้งานได้อย่างมีประสิทธิภาพ (คล้ายกับอาหารกระป๋องสำเร็จรูป...เปิดกระป๋องโปรแกรมสำเร็จรูปรับประทานได้ทันที...) ส่วนใหญ่จะถูกจัดเตรียมขึ้นมาเพื่อให้เหมาะกับการใช้งานเฉพาะอย่าง ในที่นี้จะขอกล่าวถึงโปรแกรมสำเร็จรูปซึ่งจะทำหน้าที่ในการเก็บรวบรวมข้อมูลให้เป็นไฟล์ที่มีระเบียบง่ายต่อการใช้งาน และช่วยทำให้ผู้ใช้ประมวลผลข้อมูลต่าง ๆ เหล่านั้น ตามความต้องการได้อย่างรวดเร็ว ระบบที่กล่าวถึงนี้เราเรียกว่า "Data Base Management System" ซึ่งจะขอเรียกย่อ ๆ ว่า "DBMS"

ระบบ DBMS นี้เป็นซอร์ฟแวร์ (โปรแกรม) ซึ่งจะพยายามรวบรวมความสามารถต่าง ๆ ทางด้านการสร้างไฟล์ ไฟล์ต่าง ๆ ซึ่งมีข้อมูลที่มีความสัมพันธ์กันบ้างนั้นสามารถที่จะนำมารวมกันเป็นไฟล์ใหญ่ไฟล์เดียวกันได้ ด้วยวิธีการจัดระบบข้อมูลนั้นให้เป็นระเบียบเรียบร้อย สามารถที่จะค้นหาข้อมูลที่ต้องการได้สะดวกรวดเร็ว และยังหาวิธีการที่จะประหยัดเนื้อที่ของหน่วยความจำซึ่งอาจจะใช้ "ดิสค์" เป็นเนื้อที่เก็บข้อมูลของไฟล์ต่าง ๆ ให้ประหยัดที่สุด ดังนั้นเมื่อเราใช้ DBMS เข้ามาช่วยในงานของเราแทนระบบที่ยุ่งยากของการจัดไฟล์ดังในรูปที่ 2.1 จะทำให้ผู้ใช้งานได้สะดวกรวดเร็วขึ้นโดยไม่จำเป็นต้องอาศัยผู้เชี่ยวชาญทางคอมพิวเตอร์มากนัก ระบบ DBMS จะช่วยในการจัดระบบไฟล์โดยผู้ใช้เพียงแต่ใช้คำสั่งซึ่ง DBMS ได้เตรียมไว้แล้ว เช่น เมื่อต้องการจะสร้างไฟล์ข้อมูล การเพิ่มหรือลดข้อมูล ก็สร้างใช้ได้อย่างรวดเร็วในรูปที่ 2.2 แสดงถึงหลักการของระบบ DBMS เพื่อที่จะประมวลผลข้อมูลให้ได้ตามความต้องการ



รูปที่ 2.2 ลักษณะหลักการพื้นฐานของระบบ Data Base Management System

ข้อมูลที่เก็บรวบรวมไว้ภายในไฟล์จะถูกจัดสรรด้วยการควบคุมของ DBMS ไฟล์ต่าง ๆ เหล่านั้นอาจจะถูกสร้างขึ้นพร้อมกันครั้งเดียว โดยผู้ใช้เป็นผู้ป้อนข้อมูลเข้าไปในเครื่องคอมพิวเตอร์ แล้วเราก็สามารถเขียนโปรแกรมเพื่อนำข้อมูลต่าง ๆ เหล่านี้มาประมวลผลตามความต้องการด้วยชุดคำสั่งอย่างง่าย ๆ ซึ่งระบบ DBMS ได้เตรียมไว้แล้ว ตามรูปที่ 2.2 นี้ได้เก็บรวบรวมข้อมูลเกี่ยวกับเงินเดือน บัญชี และสินค้าคงคลังไว้ภายในดิสค์ ซึ่งรวมอยู่ภายในไฟล์ชุดเดียวกัน แล้วเขียนโปรแกรมเพื่อคิดเงินเดือน ทำบัญชี และสินค้าคงคลัง โดยที่โปรแกรมต่าง ๆ เหล่านี้จะติดต่อกับข้อมูลภายในไฟล์โดยผ่านทาง DBMS การเขียนโปรแกรมเพื่อประมวลผลตามจุดประสงค์ใหม่ก็เพียงแต่เขียนโปรแกรมเพิ่ม

ขึ้น โดยอาจจะไม่ต้องสร้างไฟล์ข้อมูลใหม่ก็ได้ ซึ่งต่างจากหลักการของระบบไฟล์ธรรมดาในรูปที่ 2.1 ที่เมื่อจะเขียนโปรแกรมใหม่ก็ต้องสร้างไฟล์ใหม่ขึ้นมา ซึ่งยุ่งยากพอสมควร และแม้แต่การแก้ไขเพิ่มเติมข้อมูลในระบบ DBMS ก็จะสามารถทำได้ง่ายกว่าโดยไม่กระทบกระเทือนต่อโปรแกรมด้วย

ชนิดของระบบ DBMS

ระบบ DBMS นั้นมีวิธีการต่าง ๆ มากมายเพื่อที่จะทำให้เกิดประโยชน์อย่างน้อย 2 ประการ คือ ประหยัดเนื้อที่หน่วยความจำที่สุด และเวลาที่ใช้ในการติดต่อกับข้อมูลในไฟล์เร็วที่สุด

ระบบ DBMS แบ่งออกเป็น 2 แบบใหญ่ ๆ คือ

1. hierarchal system

และ 2. relational Data Base Management System

ความแตกต่างของทั้ง 2 แบบนี้ขึ้นอยู่กับวิธีการเก็บข้อมูลในไฟล์เป็นส่วนใหญ่กล่าวคือ hierarchal system เป็นระบบ DBMS ซึ่งมีวิธีการเก็บข้อมูลในไฟล์ที่ค่อนข้างจะยุ่งยากซับซ้อน ความสัมพันธ์ของข้อมูลแต่ละตัวนั้นจะถูกกำหนดโดยการใช้ *เซท (set)* หรือ *ตัวชี้ (pointer)* ซึ่งจะคอยบอกว่าข้อมูลแต่ละชุดนั้นมีความสัมพันธ์เรียงลำดับกันอย่างไร หรือคอยบอกว่าข้อมูลตัวถัดไปที่จะติดต่อได้นั้นจะอยู่ที่ไหน เราสามารถที่จะอ่านข้อมูลต่าง ๆ ภายในไฟล์ได้อย่างรวดเร็ว โดยการดูจากเซทซึ่งบอกลำดับ หรือดูจากตัวชี้ว่าควรจะอ่านข้อมูลนั้นจากที่ไหน

ส่วนแบบ relational DBMS เป็นวิธีการแบบที่ง่ายที่สุด และเป็นแบบที่ใช้กันโดยทั่วไป ข้อมูลต่าง ๆ เหล่านั้นถูกเก็บในลักษณะเป็นตารางจะขอยกตัวอย่างเป็นตามรูปที่ 2.3

Col.1	Col.2	Col.3	Col.4	Col.5
Invoice Number	Supplier	Description	Amount	Number
236431	Fantarac co.ltd	Prints	23.00	BBQ-747
237344	Datamat	Liths Plates	397.42	TFS-90
237345	Usa Panish	Shipping	97.00	SPT-233

รูปที่ 2.3 ตัวอย่างการเก็บข้อมูลแบบ relational DBMS

จากรูปที่ 2.3 ข้อมูลที่มองเห็นในแต่ละบรรทัดเราจะเรียกว่า *เรคอร์ด* และแต่ละบรรทัดนั้นถูกแบ่งออกเป็นหลาย ๆ คอลัมน์ แต่ละคอลัมน์นั้นเราเรียกว่า *ฟิลด์* จะสังเกตเห็นว่าข้อมูลที่อยู่ภายในตารางแต่ละช่องนั้น อาจจะเป็นข้อความหรือตัวเลขซึ่งมีความหมาย 1 อย่าง ข้อมูลในคอลัมน์เดียวกันจะเป็นข้อมูลในลักษณะเดียวกัน กล่าวคือถ้าเป็นชื่อบริษัทก็เป็นชื่อซึ่งเป็นตัวอักษรเหมือนกัน หรือถ้าคอลัมน์นั้นเป็นตัวเลขก็จะเป็นตัวเลขทั้งคอลัมน์ และที่สำคัญก็คือ ข้อมูลในแต่ละคอลัมน์เรามิได้สนใจความสำคัญก่อนหลัง เราอาจจะสลับทั้งคอลัมน์ที่ 4 และที่ 5 ได้ โดยไม่ได้ทำให้ความหมายผิดไป เป็นต้น

โปรแกรมสำเร็จรูปซึ่งใช้เป็นระบบ "dBASE II" ซอร์ฟแวร์ตัวนี้ใช้กับเครื่องไมโครคอมพิวเตอร์ซึ่งมีโปรแกรมควบคุมการปฏิบัติงาน (operating system) ของ CP/M ซึ่งเป็นตัวที่ใช้กันมากสำหรับเครื่องไมโครคอมพิวเตอร์ที่ใช้ ซีพียูZ 80 dBASE II เป็นระบบการเก็บรวมข้อมูลที่มีประสิทธิภาพมากและเป็นแบบ relational DBMS

ประโยชน์ของ dBASE II

dBASE II จะช่วยทำให้ข้อมูลต่าง ๆ มีความสับสนน้อยที่สุด โดยพยายามที่จะไม่ให้มีส่วนซ้ำกันของข้อมูลโดยไม่จำเป็น การค้นหาข้อมูลภายในไฟล์จะใช้คำสั่งอย่างง่าย ๆ การใช้งานเราสามารถทำได้โดยเริ่มจากการสร้างไฟล์ของข้อมูลที่ต้องการจะนำมาประมวล โดยใช้ชุดคำสั่งซึ่ง dBASE II ได้เตรียมไว้อยู่แล้ว คือคำสั่ง CREATE เพียงคำสั่งเดียวเราสามารถที่จะกำหนดโครงสร้างของข้อมูลในแต่ละเรคอร์ได้ตามความต้องการ จะขอยกตัวอย่างเช่น ถ้าเราต้องการสร้างไฟล์ซึ่งจะทำการเก็บ ชื่อ และที่อยู่ของบุคคลซึ่งอาจจะเป็นสมาชิกในสมาคม หรือของคนไข้ในคลีนิค ดังแสดงในรูปที่ 2.4

```
  DISPLAY   STRUCTURE
STRUCTURE FOR LIST:   B:TEST 12.DBF
NUMBER OF RECORDS:
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD              NAME          TYPE  WIDTH        DEC
001            NAME              C     015
002            SURNAME           C     020
003            ADDRESS           C     040
004            PROVINCE          C     015
005            SEX               C     006
006            BIRTHDAY          C     008
007            OCCUPATION        C     010
008            AGE               N     002
**   TOTAL     **                    00117
```


รูปที่ 2.4 ตัวอย่างโครงสร้างของไฟล์ข้อมูลที่เกี่ยวกับบุคคล (โดยใช้ dBASE II)

ผู้ใช้สามารถกำหนดชื่อของฟิลด์ต่าง ๆ ได้ตามความต้องการ เพื่อจะเก็บ ชื่อ นามสกุล ที่อยู่ จังหวัด เพศ วันเกิด อาชีพ และอายุ หรืออื่น ๆ ได้แล้วแต่จุดประสงค์

คำสั่งแต่ละคำสั่งซึ่งมีใช้อยู่ใน dBASE II นั้น ถ้าจะเขียนขึ้นมาเองด้วยภาษาคอมพิวเตอร์ทั่ว ๆ ไป เช่น ภาษาเบสิค อาจจะต้องใช้เวลาในการเขียนหลายวันทีเดียว และยังได้โปรแกรมที่ใหญ่โต ความเร็วในการทำงานก็อาจจะไม่ดีนัก เพื่อจะให้ได้ความสามารถใกล้เคียงกัน เช่น คำสั่ง CREATE ดังกล่าวมาแล้วสำหรับการเริ่มต้นสร้างไฟล์ เพียงแต่บอกโครงสร้างของไฟล์ว่าแต่ละฟิลด์ของเรคอร์ดนั้น เราตั้งเป็นชื่อตัวแปรอะไรบ้าง ชนิดของตัวแปรเป็นแบบตัวอักษร หรือตัวเลข ความยาวของแต่ละฟิลด์เป็นเท่าไร เป็นต้น หลังจากนั้นแล้ว เราก็สามารถที่จะป้อนข้อมูลซึ่งเป็นชื่อ ที่อยู่ เพศ และอาชีพ ตามที่เราได้กำหนดโครงสร้างไว้แล้วนั้น เมื่อเราต้องการจะแก้ไขข้อมูล หรือต้องการจะเพิ่มเติมข้อมูลเข้าไป dBASE II ก็ได้เตรียมชุดคำสั่งเพื่อทำตามจุดประสงค์ต่าง ๆ เหล่านั้นไว้แล้วเป็นคำสั่งว่า INSERT และ APPEND เป็นต้น

หลังจากได้ทำการป้อนข้อมูลลงในระบบ "database" แล้วเราก็สามารถที่จะทำการประมวลข้อมูลเหล่านั้นตามความต้องการ เช่น เราต้องการจะให้คอมพิวเตอร์ดึงข้อมูลของเฉพาะ ชื่อและที่อยู่ในไฟล์

นั้นมาทำการพิมพ์ออกทางเครื่องพิมพ์ เพื่อที่จะส่งจดหมายหรือต้องการจะเรียงลำดับข้อมูลของไฟล์ทั้งไฟล์นั้นใหม่ อาจจะทำการเรียงข้อมูลโดยใช้ชุดคำสั่งว่า "SORT" เพื่อจะให้ได้รายงานออกมาทางเครื่องพิมพ์ตามความต้องการได้อีกด้วย นอกจากนั้นยังมีคำสั่งพิเศษอื่น ๆ อีกมากมาย ซึ่งผู้ใช้สามารถศึกษาวิธีการใช้งานได้ง่ายจากคู่มือวิธีการใช้

Name	Sername	Address	Province	Sex	Birthday	Occulation	Age
Chanwis	Jindapoo	4 Pranchrarad 1 Rd.	Bangkok	Male	21/10/62	Student	21
Jrarat	Pingclasai	1331/4 Chun Rd. Yannava	Bangkok	Male	10/03/62	Student	20
Santi	Jitveera	19/100 Sukhapibal Rd.	Bangkok	Male	19/07/61	Student	21
Sawat	Patanasantichai	161/108 Petkasem Rd.	Bangkok	Male	05/05/61	Student	22

รูปที่ 2.5 ตัวอย่างข้อมูลที่เก็บไว้ภายในดิสค์โดยใช้ DBMS

ข้อมูลในแต่ละบรรทัด เป็นข้อมูลของบุคคล 1 คน ข้อมูลในแต่ละบรรทัดนี้เราเรียกว่า "เรคอร์ด" โครงสร้างของข้อมูลในแต่ละเรคอร์ดจะถูกแบ่งเป็นฟิลด์เหมือนกันหมด ในที่นี้เป็นฟิลด์ที่บรรจุชื่อ ที่อยู่ จังหวัดฯลฯ ตามที่ได้กำหนดโครงสร้างไว้ตามรูปที่ 2.4

จะเห็นได้ว่าระบบ Data Base Management System หรือ DBMS สำหรับเครื่องไมโครคอมพิวเตอร์นั้นเริ่มเข้ามามีบทบาทสำคัญอันหนึ่ง เหมาะที่จะใช้กับธุรกิจ ระบบงานในสำนักงาน หรือเก็บข้อมูลภายในบ้าน ที่ซึ่งมีข้อมูลที่ไม่มากนัก และโดยเฉพาะอย่างยิ่งเมื่อเครื่องไมโครคอมพิวเตอร์นับว่าจะมีประสิทธิภาพดีขึ้นและราคาถูกลง ดังนั้นระบบ DBMS นี้ก็จะเป็นระบบที่น่าจะเป็นประโยชน์ต่อผู้ใช้มากพอสมควรทีเดียว

```
.list off for age = 21
Chanwis     Jindapoo     4 Pranchrarad  1 Rd.    Bangkok   Male   21/10/62  Student  21
Santi       Jitveera     19/100 Sukhapibal Rd.   Bangkok   Male   19/07/61  Student  21
```

รูปที่ 2.6 ข้อมูลต่าง ๆ ซึ่งเราได้บรรจุอยู่ภายในแฟ้มข้อมูลบุคคลนั้น อาจจะเป็นข้อมูลของพนักงาน 200 - 300 คน เราอาจจะให้คอมพิวเตอร์ทำการสำรวจดูว่ามีบุคคลซึ่งมีอายุ 21 ปีกี่คน มีใครบ้าง ให้พิมพ์ผลลัพธ์ออกมา

3

หลักการเขียนโปรแกรมโครงสร้าง

ก่อนที่จะเรียนรู้ไมโครคอมพิวเตอร์ เรามักจำเป็นต้องศึกษาการเขียนคำสั่งโปรแกรมเรียงกันเป็นลำดับที่เรียกว่าโปรแกรม โปรแกรมจึงมักเป็นเสมือนระบบที่จะเข้าไปมีบทบาทต่อการใช้งานคอมพิวเตอร์ การเขียนโปรแกรมนั้นจำต้องมีการเข้าใจเรียนรู้ภาษาทางคอมพิวเตอร์ ภาษาคอมพิวเตอร์ก็มีมากหลายสิบภาษาโดยมีวิวัฒนาการกันมาเป็นระยะเวลายาวนาน โดยในยุคแรก ๆ ภาษาที่คุ้นเคยและใช้กันมากได้แก่ ฟอร์แทรน โคบอล อัลกอร์ เบสิค และเมื่อมีไมโครคอมพิวเตอร์เกิดขึ้นเบสิคก็เข้ามามีบทบาทอีกครั้ง ทั้งนี้เพราะตัวแปลภาษาเบสิค (Interpretter) มักจะมีขนาดไม่ใหญ่โตนัก จึงทำให้ภาษาเบสิคดูจะแพร่หลายไปมากพอสมควร แต่อย่างไรก็ตามภาษาเบสิคเป็นภาษาที่ต้องมีเลเบล หรือหมายเลขประจำทุกบรรทัด ซึ่งก็ต้องมีการอ้างถึงได้ทุกเมื่อ ดังนั้นภาษาเบสิคจึงมีโครงสร้างที่มีคำสั่ง GOTO อยู่มาก ทำให้ความยุ่งยากต่อการเขียนโปรแกรมมีมากพอควร จึงมีผู้ได้พยายามคิดหาทางออกแบบภาษาที่เป็นโครงสร้าง (structure) ซึ่งมีข้อดีในการเขียนโปรแกรมตรวจสอบ แก้ไข บำรุงรักษาได้ง่าย ภาษาในระดับไมโครคอมพิวเตอร์ที่เป็นภาษาโครงสร้าง เช่น ปาสคาล, ฟอร์ท (FORTH) ซี (C) พีแอลเอ็ม (PL/M) เป็นต้น และยังมีอีกภาษาหนึ่งที่มีแนวโน้มในการใช้งานทางธุรกิจในระบบจัดการฐานข้อมูลที่ดียิ่งในไมโครคอมพิวเตอร์ที่เป็นภาษาโครงสร้างซึ่งได้แก่ ดีเบสทู (dBASE II) ซึ่งด้วยเหตุนี้เองจึงจำเป็นต้องเข้าใจการเขียนโปรแกรมโครงสร้างกันเสียก่อน

หลักการโปรแกรม

ในการทำการงานใดก็ดีหากผู้ที่ทำงานอย่างมีระบบต้องมีการวางแผนงานจัดรูปแบบโปรแกรม แผนงานดำเนินงาน ควบคุมงานให้ได้ตามแผน และหากมีข้อบกพร่องในแผนก็ต้องบำรุงรักษาเช่นกัน

การทำงานที่มีขั้นตอนซึ่งเป็นวิธีการทางคอมพิวเตอร์ เช่น ถ้าต้องการสร้างบ้านสักหลังก็คงต้องเริ่มต้นด้วยดูจุดประสงค์ของเจ้าของบ้านว่าต้องการอะไรเพียงไร เมื่อได้จุดประสงค์แล้วสถาปนิกก็จะกำหนดแบบแปลน และวิศวกรก็คำนวณโครงสร้าง จากนั้นก็มีการก่อสร้างดำเนินการควบคุม โดยมีลำดับการทำงานเรียงกันตามลำดับ

การเขียนโปรแกรมคอมพิวเตอร์จึงจัดได้ว่ามีความสำคัญเพื่อให้การทำงานได้ผล หรือประสิทธิภาพดีที่สุด

หลักการโปรแกรมโครงสร้าง

โปรแกรมที่เขียนกันส่วนใหญ่จะมีการวางรูปแบบจากบนลงล่างหรือที่เรียกว่า top-down ผู้เขียนโปรแกรมส่วนใหญ่มักคุ้นเคยกับวิธีการเขียนโปรแกรมในรูปแบบจากบนลงล่างอยู่แล้ว แต่มักจะมีการเขียนที่อาศัยหลักการความเคยชินหรือสิ่งที่เคยทำมาแล้วเป็นรากฐาน และการเปลี่ยนแปลงรูปแบบการเขียนบางครั้งก็ทำได้ยาก ซึ่งแตกต่างจากผู้ที่เริ่มเรียนรู้ใหม่ ๆ สามารถเขียนโปรแกรมหรือเรียนรู้ในหลักการที่ถูกต้องตั้งแต่เริ่มต้นและจะทำให้มีโอกาสเป็นผู้เขียนโปรแกรมที่ดีได้ ก่อนอื่นลองดูตัวอย่างแบบการเขียนโปรแกรมที่อยู่ในรูปแบบที่ง่ายต่อการตรวจสอบดูบ้าง

ตัวอย่างข้างล่างเป็นโปรแกรมที่เขียนขึ้นเพื่อใช้ในการหาค่าสูงสุด ต่ำสุด และค่าเฉลี่ยจากข้อมูลชุดหนึ่งที่เก็บไว้ในไฟล์ชื่อ data โดยข้อมูลชุดนี้อาจเป็นตัวเลข n ตัวก็ได้

```
erase
set talk off
clear
@ 10,10 say "Average value program"
use b:data
skip
store sum to max,min
do while .not.eof
    if sum>max
        store sum to max
    else
        if sum<min
            store sum to min
        endif
    endif
    skip
enddo
goto top
count for sum>=0 to number
sum sum to total
store total/number to avr
erase
set talk on
@ 6,5 say "Maximum score is" get max
@ 7,5 say "Minimum score is" get min
@ 8,5 say "Average value is" get avr
```

ตัวอย่างที่เขียนนี้เขียนในรูปแบบโปรแกรมโครงสร้างของภาษาดีเบสทูที่มีรูปโครงที่ง่ายต่อการเข้าใจ แต่อย่างไรก็ตามลักษณะของโปรแกรมโครงสร้างจำเป็นต้องมีหลักการและรายละเอียดต่าง ๆ ประกอบซึ่งจะได้ทำการศึกษาต่อไป

หลักสี่ประการต่อการเขียนโปรแกรมที่ดี

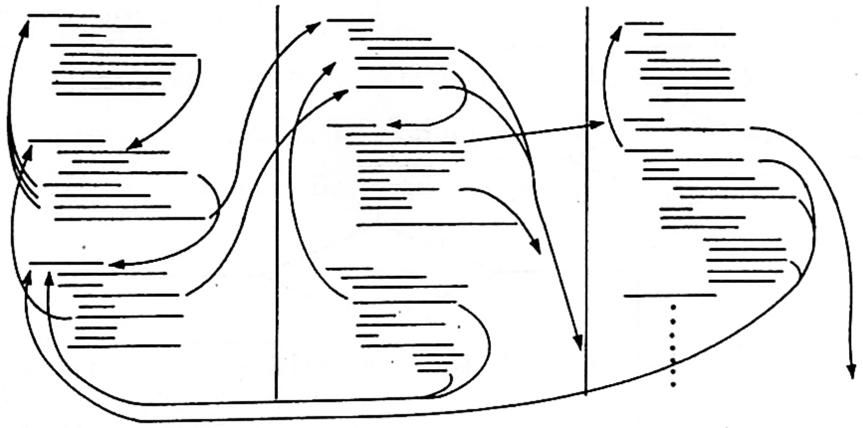
เพื่อให้การเขียนโปรแกรมสั่งงานไมโครคอมพิวเตอร์ได้ดี พึงระลึกถึงหลัก 4 ข้อนี้ไว้ในใจหรือจำให้ขึ้นใจเสียก่อน

1. ใช้ชื่อตัวแปรที่ตั้งไว้หรือชื่อข้อมูลให้มีความหมายตรงกับที่ต้องการมากที่สุด เช่น CHECK - KEY, Account-Re, Payroll ฯลฯ

2. ข้อมูลในตาราง (table) จะได้รับการออกแบบสร้างขึ้นมาเพื่อใช้เฉพาะงานอย่างเดียว จะไม่มีการใช้ตารางเพียงตารางเดียวสำหรับหลายงาน

3. ลักษณะโปรแกรมต้องเป็นการเขียนจากบนลงล่าง มีจุดเข้าเพียงจุดเดียว (1 input) และจุดออกเพียงจุดเดียว (1 output) เท่านั้น หลีกเลี่ยงการใช้คำสั่ง GOTO หรือ JUMP โดยไม่มีเงื่อนไข

4. พยายามให้คำสั่งที่เขียนชัดเจน และใช้หลักการเขียนที่ชาญฉลาดประกอบด้วย อย่าลืมว่าการเขียนโปรแกรมที่ดีอาจทำให้โปรแกรมยาวขึ้นอีกเล็กน้อย หรือทำงานได้ช้าลงอีกเล็กน้อย ก็ยังดีกว่าที่ต้องเสียเวลาในการตรวจสอบและแก้ไขโปรแกรม พึงระลึกเสมอว่าเสียเวลาในตอนแรก และไม่มีปัญหาในตอนหลังเป็นสิ่งที่น่าสนใจหรือพึงกระทำมากกว่า



โปรแกรมโครงสร้างและผังงานมีไว้ทำไม

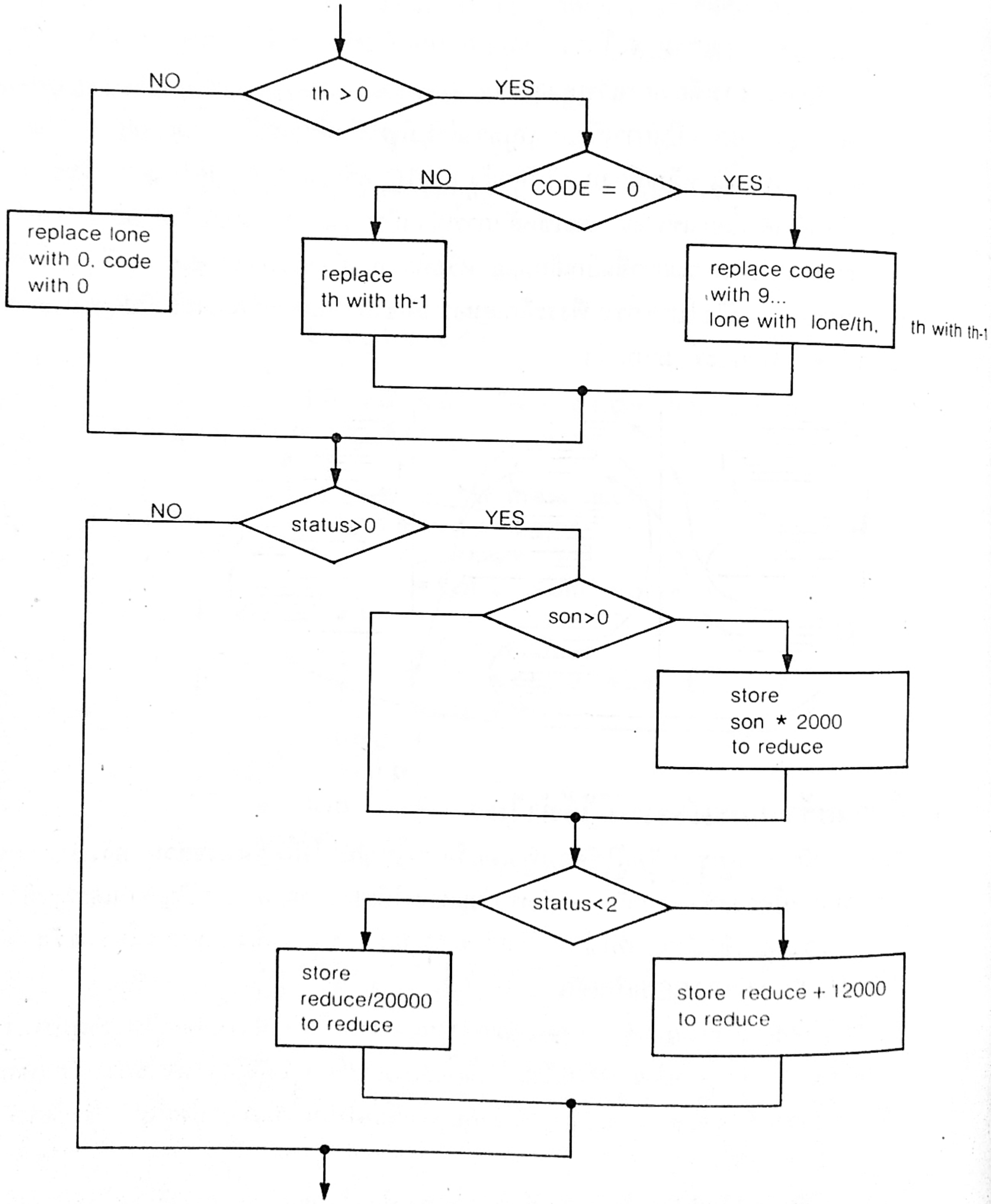
เมื่อก่อนนี้นักโปรแกรมหรือผู้ใช้คอมพิวเตอร์ส่วนใหญ่มักไม่ใคร่สนใจส่วนของโปรแกรมหรือวิธีการทางโปรแกรมโครงสร้างมากนัก แต่เมื่อพบปัญหาจึงได้พยายามหาทางแก้ปัญหา และทางออกทางหนึ่ง คือลักษณะรูปแบบของภาษาที่ใช้ หลายภาษาได้ออกแบบมาให้มีโปรแกรมโครงสร้างที่ดี เช่น PL1, PASCAL, PL/M,dBASE IIเป็นต้น

อย่างไรก็ดีภาษาที่ใช้ในการสั่งงานคอมพิวเตอร์หลายภาษาไม่สามารถจัดอยู่ในรูปแบบโปรแกรมโครงสร้างได้ง่าย เช่น FORTRAN, BASIC และแม้แต่ ASSEMBLY จึงมีผู้กำหนดวิธีการเขียนหลักการขึ้นด้วยโปรแกรมโครงสร้างและสามารถประยุกต์เข้ากับการเขียนโปรแกรมภาษาใดก็ได้ โดยยึดหลักดังนี้

1. เขียนในลักษณะใกล้เคียงกับภาษาเขียนธรรมดาเพื่ออธิบายการทำงานหรือโครงสร้างของโปรแกรม การใช้ภาษาเขียนนี้บางครั้งก็ไม่สามารถนำไปใช้ RUN กับเครื่องใดได้โดยตรงแต่สามารถเปลี่ยนรูปโปรแกรมไปในภาษาใดก็ได้โดยง่าย

2. โปรแกรมที่เขียนด้วยภาษาเขียนปกตินี้เรียกว่า pseudo code โดยผู้เขียนอาจไม่จำเป็นต้องรู้จักภาษาใดมาเลยก็ได้ ก็สามารถเขียนโปรแกรมนี้ได้

3. การเขียนในลักษณะนี้จะทำให้รูปแบบโปรแกรมเป็นโปรแกรมโครงสร้างได้ และทำให้ง่ายต่อการตรวจแก้ดัดแปลง หรือต่อเติมในภายหลังได้โดยง่าย



รูปที่ 3.2

4. สามารถแลกเปลี่ยนหรือส่งต่อให้ผู้สันทัดทางภาษาต่างภาษากันดูได้ทันที

แต่ลักษณะเขียนด้วย pseudo Code นี้จำเป็น การเขียนที่ค่อนข้างจะง่าย แต่มีโครงสร้างที่ง่ายต่อการเข้าใจการทำงานของโปรแกรม

แต่สำหรับการเขียนในหลักการของภาษาที่เป็นโปรแกรมโครงสร้างแล้ว สามารถนำคำสั่งต่าง ๆ เหล่านั้นมาเขียนได้เลยดังเช่น การเขียนโปรแกรมด้วยภาษา dBASE II นั้น รูปแบบการเขียนมีลักษณะคล้ายคำพูดธรรมดามากอยู่แล้ว

การจัดแบ่งระดับกันจะใช้วิธีการย่อหน้าเป็นชั้น ๆ โดยแต่ละการย่อหน้าจะบอกถึงระดับความลึกของส่วนของโปรแกรมนั่นเอง ลองค่อย ๆ พิจารณาจากโปรแกรมที่จะยกตัวอย่างต่อไป

รูปแบบโปรแกรมจากบนลงล่าง

เมื่อกล่าวถึงการเขียนโปรแกรมที่ดีแล้วลองพิจารณาลักษณะการวางผังโปรแกรมที่ดีบ้าง โครงร่างหลักอย่างหนึ่งที่กล่าวแล้ว คือโครงร่างการวางผังจากบนลงล่างลองพิจารณาโครงร่างเบื้องต้น ของตัวอย่างรูปที่ 3.2 ประกอบเพื่อเป็นแนวความคิด

โครงร่างโปรแกรมดังกล่าวนี้เป็นโครงร่างของโปรแกรมการคำนวณภาษีรายได้บุคคลธรรมดาในการทำบัญชีจ่ายเงินเดือน โดยโปรแกรมดังกล่าวเป็นการตัดส่วนของโปรแกรมมาเพียงนิดเดียว เพื่อให้เห็นโครงร่างการจัดทำแบบบนลงล่างตามผังงาน สิ่งที่น่าสังเกตคือโปรแกรมที่จะเขียนนี้ไม่มีส่วนใดที่ย้อนขึ้นจากล่างไปบนเลย โครงสร้างดังกล่าวเมื่อเขียนด้วยภาษา dBASE II ได้ดังนี้

```
if th>0
  if code=0
    replace code with 9,loan with loan/th,th with th-1
  else
    replace th with th-1
  endif
else
  replace loan with 0,code with 0
endif
if status>0
  if son>0
    store son*5000+study*2000 to reduce
  endif
  if status<2
    store reduce+12000 to reduce
  else
    store reduce/2.0000 to reduce
  endif
endif
```

รูปแบบของโปรแกรมที่ถูกหลักจะต้องเขียนจากบนลงล่าง และเมื่อเป็นโปรแกรมโครงสร้างก็ไม่จำเป็นต้องมีเลเบล การเขียนจะอาศัยระดับความลึกของโปรแกรมเป็นตัวบอกในแต่ละระดับที่ซ้อนกันอยู่ โปรแกรมที่เขียนด้วยหลักการทางโครงสร้างนี้จะง่ายต่อการเขียน การตรวจสอบ การดูหลักการหรือการทำงานของโปรแกรม

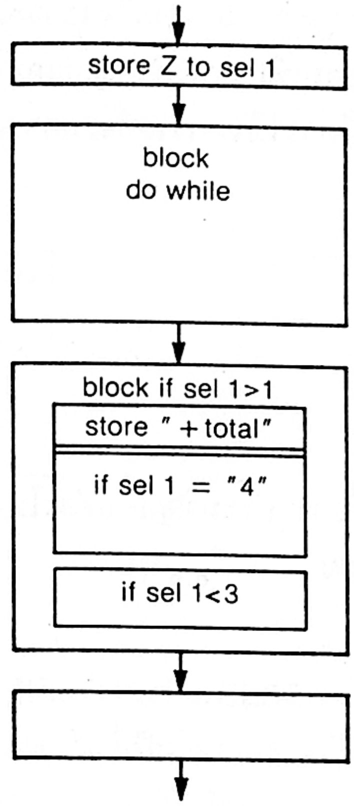
```
if th>0
   if loan=0
      replace loan with net:loan/th
   else
      replace net:loan with net:loan-loan,th with th-1
   endif
else
   replace loan with 0
endif
if status>0
   if son>0
      store son*5000+study*2000 to reduce
   endif
   if status<2
      store reduce+12000 to reduce
   else
      store reduce/2.0000 to reduce
   endif
endif
```

สัญลักษณ์ของผังงาน และการเขียนโปรแกรม

โดยหลักการแล้วผังงานที่ใช้ในการเขียนโปรแกรมทางคอมพิวเตอร์ก็มิได้ยุ่งยากอะไรมากนัก เรามักพบเห็นหลักการเหล่านี้มาแล้วกันทั้งนั้น พิจารณาจากโปรแกรมที่ 3.1 และรูปที่ 3.3

โปรแกรมที่ 3.1 ลักษณะโปรแกรมโครงสร้าง

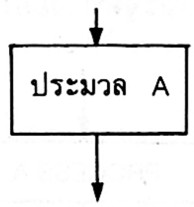
```
store "z" to sel1
  do while sel1>"4".or.sel1<"1"
  erase
  ?
  ?
  ? "        *********************************"
  ? "        *                               *"
  ? "        *          calclate  tax        *"
  ? "        *    1)    first  pay of year   *"
  ? "        *    2)    first  pay of momth  *"
  ? "        *    3)    second pay of month  *"
  ? "        *    4)    last   pay of year   *"
  ? "        *                               *"
  ? "        *********************************"
  wait to sel1
enddo
if sel1>"1"
  store "+total" to c1
endif
if sel1="4"
  store ",temp1" to c2
  store "replace tax with ttax-stax" to c4
else
  store "store temp*26 to temp1" to c5
  store "replace tax with int(ttax/26.00+temp-int(temp)+.99)" to c4
  store "+stax" to c6
endif
if sel1<"3"
  store "dialy:rate*1.2" to c3
endif
go top
do dtax
save to tt
```



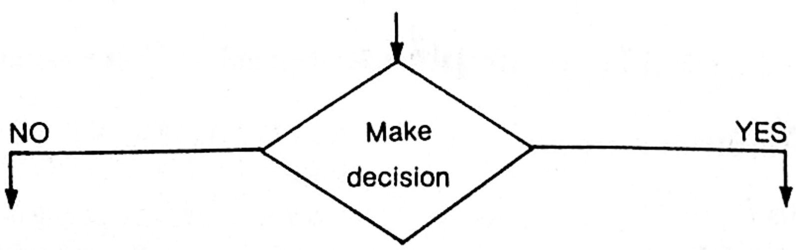
รูปที่ 3.3 การแบ่งบล็อกตามระดับจะเห็นว่าไม่มีบล็อกครอสกัน และการไหลของโปรแกรมเป็นแบบโครงสร้างบนลงล่าง

การประมวลผลส่วนใหญ่จะเขียนเป็นบล็อก หรือกล่องสี่เหลี่ยม ซึ่งโดยปกติลักษณะของผังงานเขียนได้ด้วยหลักใหญ่ ๆ ดังนี้

ส่วนการประมวลผล รูปบล็อกสี่เหลี่ยมใช้ในการประมวลผลโดยหลักการจะพยายามให้มีเพียงหนึ่งทางเข้าและหนึ่งทางออก

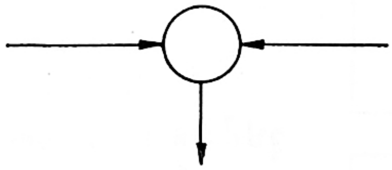


สัญลักษณ์การตัดสินใจ (if....then....else) เป็นเงื่อนไขการตัดสินใจ ถ้าเป็นไปตามเงื่อนไขก็จะไปกระทำการประมวลผลทางหนึ่ง ถ้าไม่เป็นไปตามเงื่อนไขก็ไปกระทำอีกทางหนึ่ง ในกรณีนี้มี 1 อินพุต 2 เอาต์พุต แต่รูปโครงที่จะเขียนในโปรแกรมโครงสร้างต่อไปต้องรวมให้เหลือเพียง 1 เอาต์พุต ซึ่งจะกล่าวถึงต่อไป



รูปที่ 3.4

การรวมสัญญาณ หากการไหลของโปรแกรมมาจากสองที่มารวมกัน เราจะใช้วิธีการรวมด้วยสัญลักษณ์ที่กล่าวถึงนี้ การรวมโดยหลักการมักจะรวมเพียงสองสัญญาณออกมาเป็นหนึ่งสัญญาณ กรณีนี้จะมี 2 อินพุต ออกเป็น 1 เอาต์พุต ซึ่งจะมีวิธีการใช้ในกรณีที่จะกล่าวต่อไป



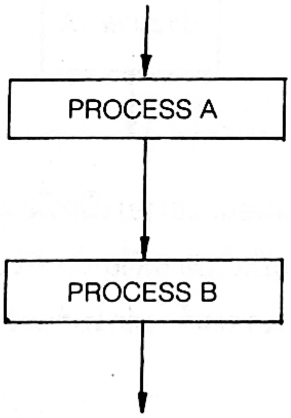
รูปที่ 3.5

เส้นต่อของแต่ละบล็อก การเชื่อมแต่ละบล็อกหรือการประมวลผล แต่ละโมดุลจะใช้วิธีการเขียนเส้นลูกศร ทิศทางของลูกศรแสดงการไหลของโปรแกรม



รูปที่ 3.6

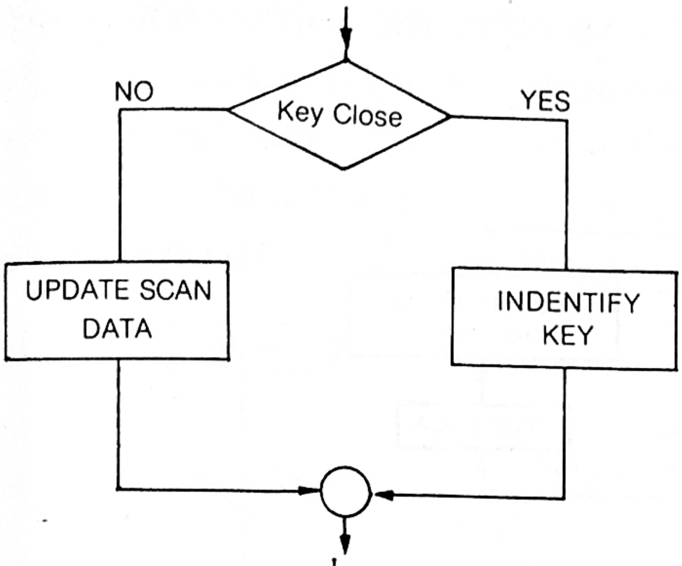
การสร้างลำดับของโปรแกรม ในการเขียนโครงสร้างโปรแกรมจำเป็นต้องจัดลำดับโปรแกรมให้ได้รูปแบบที่ถูกต้อง ง่ายต่อการเขียน การตรวจสอบและแก้ไข จึงต้องมีการเขียนในลักษณะโครงสร้างต่อเนื่อง เมื่อนำบล็อกของการประมวลผลแต่ละบล็อกเข้ามาประมวลผลซึ่งเชื่อมต่อเป็นลำดับกัน จะพยายามรักษาให้มีจุดทางเข้าหนึ่งและจุดทางออกหนึ่งเช่นเดิม



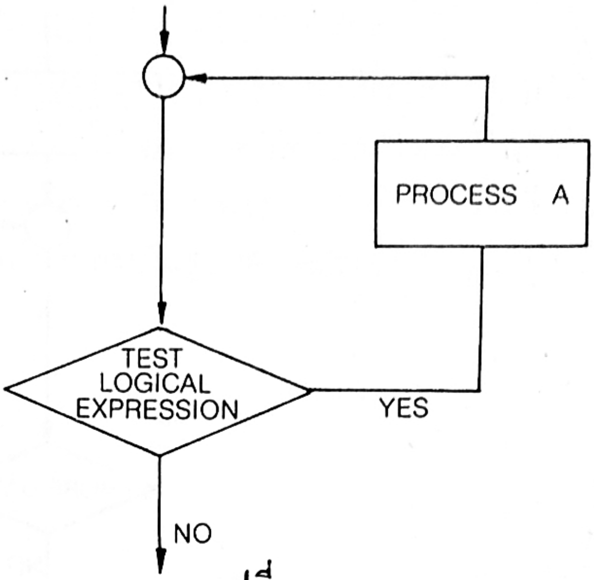
รูปที่ 3.7

เขียนในรูปโปรแกรมได้เป็น

```
do demo;
    process a
    process b
end do
```



รูปที่ 3.8



รูปที่ 3.9

โครงสร้าง IF...THEN....ELSE

โครงสร้างนี้เป็นโครงสร้างที่ใช้ในการตรวจสอบเงื่อนไขและแยกการทำงานตามเงื่อนไขที่วางไว้ เขียนในรูปโปรแกรมได้เป็น

```
do demo2;
    if key close
       indentify key
    else
       update scan data
    end if
end do
```

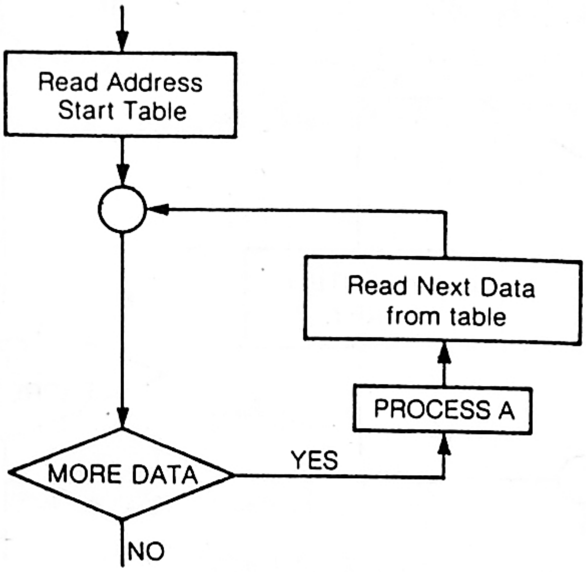
โครงสร้างแบบลูป

การทำงานของคอมพิวเตอร์แบบหนึ่ง คือ การทำงานที่ต้องทำซ้ำซ้อนวนไปมาเป็นลูป รูปแบบของโปรแกรมแบบลูปนี้จึงมีที่ใช้มากพอควร รูปแบบลูปที่ใช้กันแยกแยะเป็น 2 แบบ คือ

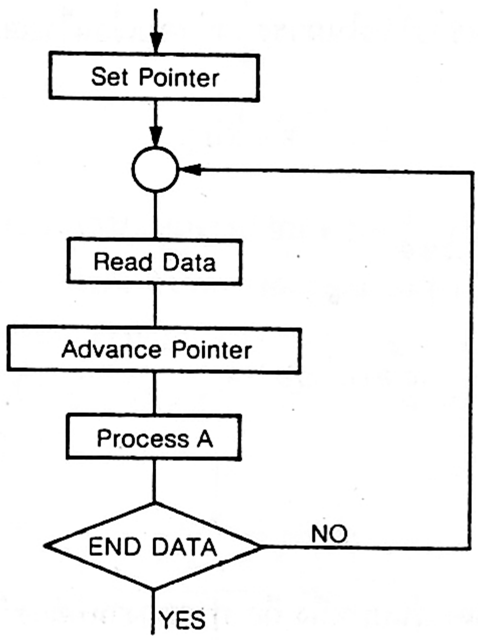
1. รูปโครง DO WHILE เป็นรูปโครงที่สร้างลูป ลองดูจากรูปโครงสร้างในรูปที่ 3.9 การประมวลผล A จะเกิดขึ้นถ้าเงื่อนไขการทดสอบเป็นจริง และตราบใดที่มีการวนมาและเงื่อนไขเป็นจริง (YES) ก็จะกระทำการลูป การหลุดจากลูปจะเกิดขึ้นเมื่อเงื่อนไขเป็นเท็จ (NO)

ลองพิจารณาโครงสร้างโปรแกรมและโปรแกรมโครงสร้างดังรูปที่ 3.10

```
table-demo-proceduce
read add-start-table
do while (more data = yes)
   process a
   read next data from table
end do
```



รูปที่ 3.10 โครงสร้าง DO WHILE



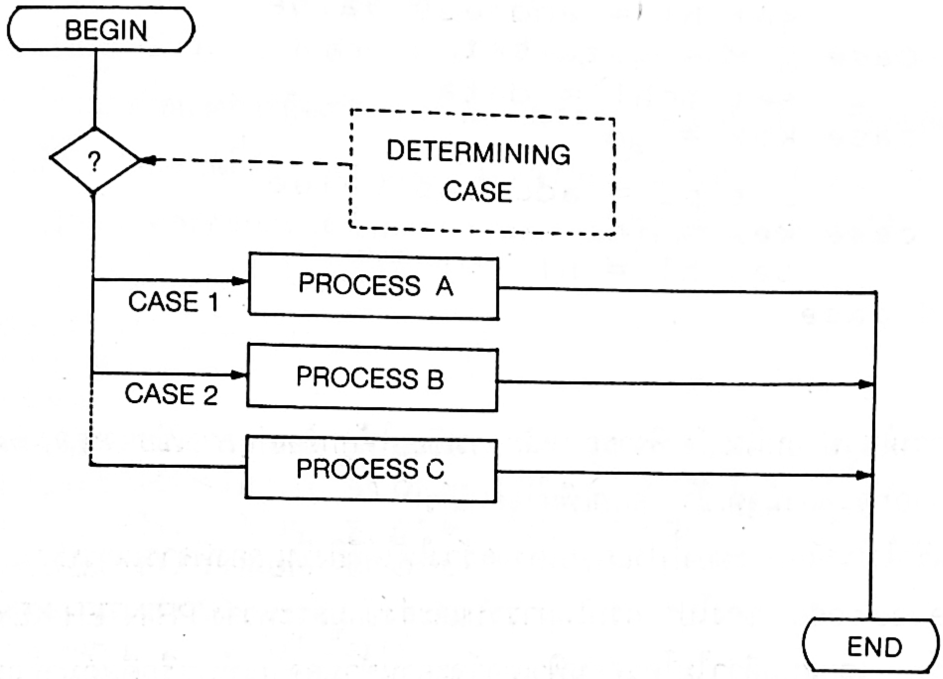
รูปที่ 3.11 โครงสร้าง DO UNTIL

2. รูปโครงของ DO UNTIL เป็นอีกรูปแบบหนึ่งของการลูป โครงสร้างแบบนี้จะทำการประมวลผลก่อนโดยทำจนกระทั่งถึงค่าที่ต้องการ เมื่อเขียนรูปโปรแกรมจะเป็นโครงสร้างดังนี้

```
demo3-proceduce
set pointer
do until (end data)
    read data
    advance pointer
    process a
end do
end demo3
```

โครงสร้างการเลือกทางใดทางหนึ่ง (SELECT หรือ CASE)

โปรแกรมที่เขียนขึ้นทุกกรณีจะสามารถแทนรูปแบบของผังงานโครงสร้างและเขียนโปรแกรมโครงสร้างขึ้นก่อนได้เสมอ ยังมีอีกโครงสร้างหนึ่งซึ่งโดยหลักการแล้วมาจากหลักการของ IF THEN ELSE ซึ่งเป็นลักษณะของการเลือกทางใดทางหนึ่งจากหลาย ๆ ทาง โดยรูปการแล้วสามารถนำ IF THEN ELSE มาใช้ได้ ลองพิจารณาการเขียนโปรแกรมในรูป IF THEN ELSE ดูก่อน



รูปที่ 3.12

โปรแกรมตัวอย่างที่กล่าวถึงนี้ เป็นโปรแกรมแสดงการตรวจค่าคีย์ฟังก์ชันดูว่าเป็นคีย์อะไร และการทำงานจะกระทำตามคีย์ฟังก์ชั่นนั้น ๆ

```
read key
if key = address set
   set hl = address value
else
     if key = data set
        set mchl = data
     else
        if key = go
           set pc = address value
        else
           if key = inc
              set hl = hl + 1
           end if
        end if
     end if
end if
```

*

ด้วยวิธีการนี้สามารถเขียนโปรแกรมให้อยู่ในรูปใหม่ได้เช่นเดียวกัน ลองพิจารณาจากตัวอย่างในการใช้ case ดูบ้าง

```
read key
do case
   case key = address set
        set hl = address value
   case key = data set
        set mchl = data
   case key = go
        set pc = address value
   case key = inc
        set hl = hl + 1
end case
```

การเขียนโปรแกรมในรูปแบบนี้จะต้องพิจารณาหลักการเขียนให้ถูกต้อง การเขียนจะต้องตัดแบ่งเป็นบล็อกๆ ได้ โดยแต่ละบล็อกจะต้องมีจุดเข้า - ออกเพียงจุดเดียว

เพื่อเข้าใจในลักษณะของโปรแกรมโครงสร้างให้ได้ดียิ่งขึ้น ลองพิจารณารูปแบบของโปรแกรมที่เขียนด้วย pseudo code ของโปรแกรมในการรับและส่งข้อมูลระหว่าง PRINTER KEYBORD + CRT กับคอมพิวเตอร์ดู โดยสมมติว่าไมโครคอมพิวเตอร์จะอ่านข้อมูลจากเทปหรือดิสค์มาทีละบรรทัดเพื่อทำการตรวจแก้ไข เมื่ออ่านจากเทปหรือดิสค์แล้วจะแสดงผลบนจอซีอาร์ที เพื่อรอรับคำสั่งเกี่ยวกับการแก้ไข ถ้าหากไม่แก้ไขก็จะพิมพ์ LF (line feed) หรือ CR (cariage return) ซึ่งไมโครคอมพิวเตอร์ก็จะอ่านข้อมูลบรรทัดต่อไปจนจบ แต่ถ้าบรรทัดใดต้องการแก้ไขโดยทำการแก้เปลี่ยนตัวให้พิมพ์ข้อความที่จะแก้จบด้วย LF ถ้าไม่จบด้วย LF แต่จบด้วยตัวอักษรอื่นจะหมายถึง การแทรกตัวอักษร ลองพิจารณาจากโปรแกรมโครงสร้างที่เขียนเป็น pseudo code

```
do until end of input text
   get line from input and type on CRT
   get response line from key board,store and
   echo if only CR or LF enter
      do nothing
   else
      if last character if LF
         do character changes and output line
      else
         do character inserts and output line
      end if
   end if
end do
```

รูปแบบโปรแกรมโครงสร้างใด ๆ ก็ตาม เมื่อเขียนขึ้นแล้วจะต้องแสดงได้เด่นชัด ในเรื่องส่วนของโปรแกรมที่สามารถแยกแยะออกเป็นส่วน ๆ ได้โดยไม่มีผล เพราะลักษณะของโปรแกรมที่อยู่ระดับเดียวกันสามารถแสดงลำดับได้ในบล็อกหนึ่ง เช่น IF....END IF ก็คือ 1 บล็อก ลองทำการเขียนจากโปรแกรมโครงสร้างให้เป็นผังงานใหม่จะเห็นได้ว่าง่ายต่อการเขียนกลับเป็นผังงานมาก

การเขียนโปรแกรมโครงสร้างในภาษาดีเบสทู

ตัวอย่างที่จะกล่าวถึงนี้เป็นตัวอย่างการเขียนโปรแกรมด้วยโปรแกรมโครงสร้างด้วยดีเบสทู ลองพิจารณาจากตัวอย่างการคิดเกรดนักเรียนโดยสมมติว่า คะแนนของนักเรียนแต่ละคนเก็บไว้ในไฟล์ที่ชื่อว่า data ถ้าทราบคะแนนที่ต้องการกำหนดเกรดนั้นว่าคนที่ได้เกรด A, B, C, D จะต้องได้คะแนนไม่น้อยกว่าเท่าไร เมื่อกำหนดคะแนนแล้วก็ทำการกำหนดเกรดให้นักเรียนแต่ละคนได้

```
type grade.cmd
        erase
        clear
        input"Minnmum score for grade A" to a
        input"Minimum score for grade B" to b
        input"Minimum score for grade C" to c
        input"Minimum score for grade D" to d
        use b:data
        do while .not.eof
             if sum>=a
                  replace grade with "A"
                  skip
                  loop
             endif
             if sum>=b
                  replace grade with "B"
                  skip
                  loop
             endif
             if sum>=c
                  replace grade with "C"
                  skip
                  loop
             endif
             if sum>=d
                  replace grade with "D"
                  skip
                  loop
             endif
             replace grade with "F"
             skip
        enddo
        return
```

ดีเบสทูจึงเป็นภาษาที่นำมาใช้ในการเขียนโปรแกรมโครงสร้างในการทำงานที่เกี่ยวข้องกับฐานข้อมูลโดยตรง การเรียนรู้ดีเบสทูและโปรแกรมโครงสร้างจึงเป็นของคู่กัน สังเกตว่าการเขียนโปรแกรมโครงสร้างจะไม่มี GOTO ไม่มีเลเบล โปรแกรมจะเขียนเรียงจากบนลงล่างเสมอ การใช้ระดับความลึกเป็นวิธีการแยกให้เกิดโครงสร้างทำให้ง่ายต่อการอ่านโปรแกรมโดยไม่ต้องเขียนผังงาน

4

การสร้างไฟล์ และการแก้ไขข้อมูลในไฟล์

ดีเบสทู เป็นภาษาที่ใช้ในการจัดการระบบข้อมูล (Data Base Management System) ซึ่งข้อมูล (data) ที่เก็บเอาไว้จะได้รับการบันทึกในลักษณะของตารางที่มี 2 มิติ และความสัมพันธ์ของข้อมูลจะมีอยู่ระหว่างแถว (ROW) แต่ละแถว และระหว่างคอลัมน์แต่ละคอลัมน์ดังตัวอย่าง

	คอลัมน์1	คอลัมน์ 2	คอลัมน์ 3
	INVOICE NUM.	SUPPLIER	DESCRIPTION
แถว 1	2386	GRAPHIC PROCESS	PRINTS
แถว 2	1234	AIR FIRTH LTD.	SHIPPING

คำจำกัดความของศัพท์เฉพาะที่ใช้

เรคอร์ด หมายถึง แถวแต่ละแถวตามแนวขวางของตาราง

ไฟล์ หมายถึง ชื่อคอลัมน์แต่ละคอลัมน์ ใน 1 เรคอร์ดอาจมีได้หลายฟิลด์ ซึ่งแต่ละฟิลด์จะมีชื่อต่าง ๆ กันเพื่อสะดวกในการค้นหา

หมายเหตุ

1. ค่าของข้อมูลแต่ละค่าที่ใช้ในตารางต้องเป็นค่าเดียวเท่านั้น คือจะใช้ข้อมูลชุดไม่ได้ (ข้อมูลที่เรียกว่าอาเรย์)

2. ลักษณะการเก็บของข้อมูลในคอลัมน์หนึ่ง ๆ ต้องเป็นชนิดเดียวกัน เช่น เป็นตัวอักษร ตัวเลข หรือลอจิก (character, numeric or logical) โดยข้อมูลในส่วนตัวเลขเท่านั้นที่สามารถทำการคำนวณได้

3. แต่ละเรคอร์ดจะเป็นส่วนเฉพาะ ๆ ของข้อมูลในเรคอร์ดนั้นไม่เกี่ยวข้องกัน

ชนิดของข้อมูล (data types) แบ่งออกเป็น 3 ชนิดคือ

1. **ตัวอักษร** (character) หมายถึง ข้อมูลทั้งหมดที่สามารถพิมพ์ได้ในลักษณะของรหัส ASCII รวมทั้งเลขจำนวนจริง สัญลักษณ์ต่าง ๆ และช่องว่าง (space)

2. **ตัวเลข** (numeric) หมายถึง จำนวนเลขบวกหรือลบใดใด (ค่าความถูกต้อง 10 ตำแหน่ง) คิดตัวเลขถูกต้องแบบตัวเลขโฟลตติ้งพอยท์ (floating point)

3. **ข้อมูลลอจิก** (logical) หมายถึง ค่าทางตรรกศาสตร์ที่ให้ความหมายเฉพาะเป็นจริง true หรือเท็จ (false) เท่านั้น สำหรับดีเบสทูนี้กำหนดให้จริงแทนด้วย T, t, Y, y เท็จแทนด้วย F, f, N, n

การใช้ดีเบสทู

เมื่อต้องการจะใช้ดีเบสทู เพียงแต่ใส่แผ่นดิสค์ของดีเบสทูในช่องจานแม่เหล็กแล้วกดคีย์ DBASE ตามด้วย<RET>จากนั้นลักษณะที่ปรากฏบนจอภาพจะเป็นดังนี้

ENTER DATE AS MM/DD/YY OR RETURN FOR NONE:

สำหรับการที่จะใส่วัน เดือน ปี ลงไปนี้ เพื่อที่จะทำให้ทราบว่าการใช้งานนี้ใช้ครั้งสุดท้ายเมื่อไร เพื่อประโยชน์เวลาพิมพ์รายงาน (report) หากไม่ต้องการจะใส่วัน เดือน ปี ก็กดปุ่ม<RET>

ลักษณะบนจอภาพจะเป็นดังนี้

*** dBASE II VER 2.3 ***

หลังจากนั้นดีเบสทูก็เตรียมพร้อมที่จะรับคำสั่งได้แล้วโดยขึ้นสัญญาณเตรียมพร้อมเป็น . โดยทั่วไปแล้วลักษณะของคำสั่งของดีเบสทูจะคล้ายประโยคหรือคำที่คุ้นเคยอยู่แล้ว

คำสั่งในภาษาดีเบสทู

คำสั่งในการสร้างไฟล์ เพื่อให้เข้าใจเกี่ยวกับภาษาดีเบสเบื้องต้นจะทำการยกตัวอย่างของการสร้างไฟล์ในการเก็บข้อมูลซึ่งจะให้ชื่อว่า *WARRANTY* ซึ่งเป็นฐานข้อมูลสำหรับเก็บข้อมูลของบริษัทแห่งหนึ่ง ซึ่งเป็นบริษัทขายประกัน

การสร้างข้อมูลขึ้นมาใหม่นี้ให้พิมพ์คำว่า CREATE เมื่อกด <RET> แล้วผลตอบสนองบนจอภาพจะเป็นดังนี้

```
. CREATE
ENTER FILENAME: WARRANTY
```

ซึ่งในที่นี้จะใส่ชื่อของไฟล์ว่า WARRANTY

ข้อกำหนดของการตั้งชื่อไฟล์

1. อักขระแรกของชื่อไฟล์ต้องเป็นตัวอักษร
2. ใช้ชื่อไฟล์ แต่ละชื่อจะมีสูงสุดได้ 8 ตัวอักษร โดยที่ต้องไม่มีช่องว่างหรือเครื่องหมายโคลอน (COLON) ในชื่อไฟล์นั้น

เมื่อใส่ชื่อไฟล์เสร็จแล้วกด <RET>ตัวดีเบสทูจะทำการสร้างไฟล์ขึ้นมาใหม่ชื่อว่า WARRANTY.DBF ความหมายของ DBF คือ ไฟล์ฐานข้อมูลส่วนที่จะแสดงออกมาทางจอภาพ จะเป็นดังนี้

```
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD   NAME,TYPE,WIDTH,DECIMAL PLACES
 001
```

ซึ่งในตัวอย่างของไฟล์ที่จะสร้างจะมีรายละเอียดตามโครงสร้างของไฟล์ดังนี้

```
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD    NAME,TYPE,WIDTH,DECIMAL PLACES
 001      CARDNUM,C,6
 002      SERIALNUM,C,6
 003      MACHINTYPE,C,2
 004      USRCOMPY,C,15
 005      PURPRICE,N,10,2
 006      DATE,C,8
 007
```

ในแต่ละเรคอร์ดจะต้องมีรายละเอียดอยู่ 6 ชนิด (แต่ละชนิดคือรายละเอียดใน 1 ฟิลด์) ซึ่งข้อกำหนดในการกำหนดรายละเอียดของฟิลด์มีดังนี้

ชื่อฟิลด์ (field name) มีได้สูงสุด 10 ตัวอักษร (จะใส่ตัวใหญ่หรือเล็กก็ได้) แต่ละชื่อจะต้องขึ้นต้นด้วยตัวอักษร ไม่มีช่องว่าง (space) ในชื่อนั้น และอาจจะมีจุดหรือเครื่องหมายโคลอนก็ได้

ชนิดหรือ Type (Type of Data) ซึ่งใช้แทนด้วยตัวอักษรดังต่อไปนี้

C หมายถึง ข้อมูลที่เป็นตัวอักษร (character)

N หมายถึง ข้อมูลที่อยู่ในฟิลด์นั้นต้องเป็นตัวเลข (number)

L หมายถึง ข้อมูลเป็นค่าทางลอจิก

ความกว้าง Width หรือ Field Width หรือความกว้างของฟิลด์จะมีขนาดเท่าใดก็ได้ แต่สูงสุดจะมีได้ไม่เกิน 254 ตัวอักษร

แต่ถ้าข้อมูลในฟิลด์นั้นเป็นชุดของตัวเลข ที่มีจุดทศนิยมจะถือว่าจุดทศนิยมนั้นถือว่าเป็น 1 ตัวอักษร

ลองใส่รายละเอียดของข้อมูลที่แสดงไว้ข้างบน เมื่อใส่ข้อมูลในฟิลด์ที่ 6 แล้วกดคีย์ RET บนจอภาพจะเป็น 007

ให้กด RET ซ้ำอีกครั้งซึ่งหมายถึงการจบการใส่ข้อมูลของรายละเอียดของไฟล์เพียงเท่านี้และดีเบสทูจะเก็บโครงสร้างของไฟล์นี้เอาไว้ และจะมีคำถามว่าจะใส่ข้อมูลเลยหรือไม่ดังนี้

```
INPUT DATA NOW? Y
```

ให้กด Y ซึ่งหมายถึง yes คือการต้องการใส่ข้อมูลเลย ถ้าหากกด N หรือ RET เครื่องจะกลับไปยังเครื่องหมายเตรียมพร้อมรับคำสั่งคือจุด เพื่อให้ผู้ใช้ป้อนคำสั่ง

การใส่ข้อมูลลงในไฟล์ใหม่ของดีเบส

หลังจากคีย์ "Y" เข้าไปแล้วจะปรากฏเบอร์ของเรคอร์ดแรก และชื่อของฟิลด์บนจอภาพความยาวของแต่ละฟิลด์จะแสดงโดยเครื่องหมายโคลอน (:) ตำแหน่งของตัวเคอร์เซอร์จะแสดงตำแหน่งของข้อมูลที่เราจะใส่ลงไป เมื่อใส่ข้อมูลจนเต็มในฟิลด์ใด ๆ แล้วจะมีเสียงปิ๊บเกิดขึ้นและตัวเคอร์เซอร์จะลงไปใน

ฟิลด์ใหม่ทันที และสามารถที่จะเลื่อนตัวเคอร์เซอร์ไปสู่ฟิลด์ถัดไปได้โดยกดคีย์ RET และสามารถเลื่อนตัวเคอร์เซอร์กลับไปยังฟิลด์ก่อนหน้าได้โดยใช้กดคีย์ CTRL-E (กดคีย์ CTRL และ E พร้อมกัน)

สมมติว่าได้ใส่ข้อมูลต่อไปนี้ลงในแต่ละเรคอร์ด

CARDNUM	SERIALNUM	MACHINTYPE	USRCOMPY	PURPRICE	DATE
954873	998717	DU	BOON THAIYONT	90000.99	12/01/83
971981	997514	DU	SOOTSUWAN	95000.00	11/10/83
750793	767231	KB	P.NAKORN LTD.	30500.77	12/10/83
798717	765211	KB	SIAM PANISH	31000.95	11/29/83
546798	567891	PT	BOON THAIYONT	19005.95	12/10/83
875179	768811	KB	HOME ELECTRONIC	32000.91	10/05/83
557314	569811	PT	HOME ELECTRONIC	7510.98	11/29/83
558111	568791	PT	VIMONSILP LTD.	7610.98	11/29/83
951283	998111	DU	VIMONSILP LTD.	91000.88	11/05/83

เมื่อต้องการจะยุติการใส่ข้อมูลในขณะที่ ตำแหน่งของตัวเคอร์เซอร์อยู่ที่ตำแหน่งแรกของฟิลด์แรกของเรคอร์ดอันต่อไป ให้กด<RET> จากนั้นข้อมูลทั้งหมดที่ใส่เข้าไปจะถูกนำไปเก็บในไฟล์ที่อยู่บนแผ่นจานแม่เหล็กทันที

หากมีการใส่ข้อมูลผิดพลาดลงไปก็ไม่เป็นไร เพราะในรายละเอียดต่อไปมีรายละเอียดเกี่ยวกับการแก้ไขข้อมูล เมื่อต้องการจะหยุดพักหรือถอดแผ่นจานแม่เหล็กออกให้ใช้คำสั่ง QUIT เพราะคำสั่งนี้จะทำการปิดไฟล์ทั้งหมดที่กำลังใช้ในระบบฐานข้อมูลขณะนั้น หากไม่ทำเช่นนั้นอาจจะทำให้ข้อมูลที่อยู่ภายในบางส่วนเสียหายได้

การแก้ไขข้อมูลในไฟล์ฐานข้อมูล

การใช้คำสั่ง USE เมื่อเกิดข้อผิดพลาดในการใส่ข้อมูล หรือต้องการจะเปลี่ยนแปลงแก้ไขส่วนใดของข้อมูลที่ใส่ไปแล้ว สามารถทำได้โดยใช้การแก้ไขบนจอภาพด้วยการเลื่อนเคอร์เซอร์ขึ้นลงหรือซ้ายขวาเพื่อทำการแก้ (full screen editing) แต่ต้องคีย์คำว่า dBASE หรือ TBASE เพื่อทำการโหลดโปรแกรมเข้าสู่หน่วยความจำก่อน (ในกรณีที่เครื่องแสดง A >) หลังจากนั้นใช้คำสั่ง USE WARRANTY เพื่อเรียกเอาข้อมูลนั้นออกมาทำการแก้ไข หลังจากนั้นหากต้องการแก้ไขข้อมูลที่เรคอร์ดใดก็เพียงแต่พิมพ์คำว่า EDIT (n) เข้าไป เมื่อ (n) คือ record number ที่ต้องการแก้ไขนั้น

การแก้ไขบนจอภาพ (full screen editing) สมมติว่าเราต้องการแก้ไขข้อมูลในเรคอร์ดที่ 4 ในฟิลด์ที่ชื่อว่า "PURPRICE" จาก 31000.95 เป็น 30000.55 ดังนั้นให้พิมพ์คำว่า EDIT 4 จากนั้นข้อมูลของเรคอร์ดที่ 4 จะแสดงให้เห็นบนจอภาพ การแก้ไขก็เพียงแต่ใช้ CTRL X เพื่อเลื่อนตำแหน่งเคอร์เซอร์ลงมายังตำแหน่งที่ต้องการแก้ไขแล้วก็พิมพ์คำใหม่ลงไปทันที

ในขณะนี้เท่ากับว่าเรากำลังแก้ไขข้อมูลในเรคอร์ดที่ 4 อยู่ หากต้องการจะแก้ไขเพิ่มเติมให้ใช้คำสั่ง CTRL E เพื่อเลื่อนตำแหน่งเคอร์เซอร์ขึ้นไปยังฟิลด์ที่ต้องการแก้ไขใหม่ได้ทันทีโดยไม่ต้องใช้คำ

สั่ง EDIT 4 อีกครั้ง ลองใช้คำสั่ง CTRL E เพื่อเลื่อนตำแหน่งเคอร์เซอร์ไปที่ฟิลด์ชื่อว่า SERIALNUM เพื่อแก้ตัวเลขจาก 765211 เป็น 765222 จะเห็นว่าการแก้ไขตัวเลขนั้นควรจะแก้เฉพาะ 2 ตำแหน่งสุดท้ายเท่านั้น จึงควรใช้ CTRL D เพื่อเลื่อนตำแหน่งเคอร์เซอร์ให้ผ่าน 4 ตำแหน่งแรกไปทางขวาแล้วพิมพ์ตัวเลขที่ถูกต้องทับลงไป และหากเลื่อนตำแหน่งเคอร์เซอร์มากเกินไปก็สามารถเลื่อนกลับไปทางซ้ายได้โดยใช้ CTRL S

ในที่นี้ต้องการแก้ไขฟิลด์ USRCOMPY อีกก็ทำได้โดยการเลื่อนตำแหน่งเคอร์เซอร์ไปยังฟิลด์ที่ต้องการโดยใช้ CTRL E และ CTRL X จากนั้นก็แก้ไขโดยเปลี่ยนชื่อบริษัทจาก SIAM เป็น SONY ในกรณีนี้จะใช้ CTRL D เพื่อจะวางตำแหน่งของเคอร์เซอร์ให้อยู่ที่ตัว "I" ของคำว่า SIAM จากนั้นกด CTRL G 3 ครั้ง ตัวอักษรบนเคอร์เซอร์และทางขวาของตำแหน่งเคอร์เซอร์จะถูกลบออกไปทันที จากนั้นก็ให้พิมพ์คำว่า ONY ลงไปหลังคำว่า S โดยใช้ CTRL V เพื่อเริ่มขั้นตอนการแทรกข้อความ เมื่อพิมพ์คำว่า ONY หลังจากกด CTRL V แล้วตัวอักษรที่ใส่เข้าไปจะอยู่ทางด้านหน้าของตำแหน่งเคอร์เซอร์

หลังจากแก้ไขเสร็จแล้วใช้คำสั่ง CTRL R ระบบทั้งหมดจะทำการเก็บข้อมูลที่แก้ไขใหม่แล้วในเรคอร์ดเดิม และหากต้องการจะแก้ไขเรคอร์ดที่ 2 ตรง USRCOMPY จาก SIAM เป็น SONY อีกก็ใช้ CTRL R อีกครั้ง หากใช้ CTRL W จะมีความหมายเป็นการเก็บข้อมูลที่แก้ไขแล้วลงในดิสค์และกลับไปตั้งต้นใหม่ที่ดีเบสทู

แต่ถ้าหากไม่ต้องการให้ข้อมูลที่แก้ไขแล้วเก็บลงในแผ่นจานแม่เหล็กก็เพียงแต่ใช้ CTRL Q ก็จะกลับเข้าสู่ดีเบสทูทันทีโดยไม่มีการแก้ไขข้อมูล สิ่งที่ปรากฏคือจะขึ้นเครื่องหมายเตรียมพร้อมรับคำสั่งต่อไป (.)

การแก้ไขบนจอภาพโดยใช้คำสั่งเกี่ยวกับการทำงานของเคอร์เซอร์ สามารถสรุปได้ดังนี้

(CTRL X) หรือ CTRL F จะเลื่อนตำแหน่งเคอร์เซอร์ลงไปยังฟิลด์ที่ติดกัน
(CTRL E) หรือ CTRL A จะเลื่อนตำแหน่งเคอร์เซอร์ขึ้นไปยังฟิลด์ที่ติดกัน
(CTRL D) จะเลื่อนตำแหน่งเคอร์เซอร์ไปข้างหน้า 1 ตำแหน่งตัวอักษร
(CTRL S) จะเลื่อนตำแหน่งเคอร์เซอร์กลับไป 1 ตำแหน่ง
(CTRL V) จะเปลี่ยนคำสั่งที่เปลี่ยนกลับไปมาระหว่างการพิมพ์ทับและการแทรกข้อความ
(CTRL G) จะลบอักษรใด ๆ ก็ตามที่ตำแหน่งของตำแหน่งเคอร์เซอร์
<RUBOUT> จะลบข้อความทั้งหมดที่อยู่ทางซ้ายของตำแหน่งเคอร์เซอร์ออก
(CTRL P) จะเป็นเสมือนคำสั่งสำหรับเปิดเครื่องพิมพ์ หรือปิดเครื่องพิมพ์
(CTRL Q) ยุติการแก้ไขและจะกลับไปยังการใช้ดีเบสทูที่จุดแรกใหม่ โดยไม่เก็บข้อความที่ทำการแก้ไขแล้วลงในจานแม่เหล็ก ตำแหน่งเคอร์เซอร์จะไปอยู่หลังเครื่องหมายเตรียมพร้อมรับคำสั่ง (.)

หมายเหตุ คำสั่งที่อยู่ในเครื่องหมาย ( ) หมายถึงให้กดคีย์ (CTRL) ก่อน แล้วจึงกดตัวอักษรใด ๆ ที่ต้องการตามหลังโดยยังไม่ปล่อยมือจากคีย์ CTRL

สรุปฟังก์ชันสำหรับการแก้ไข

(CTRL C) จะเขียนเรคอร์ดนั้น ๆ ลงในแผ่นจานแม่เหล็กหลังจากแก้ไขแล้ว และจะเรียกเรคอร์ดต่อไปออกมาทันที

(CTRL R) จะเขียนเรคอร์ดนั้น ๆ ลงในแผ่นจานแม่เหล็กหลังจากแก้ไขแล้ว และจะเรียกเรคอร์ดที่อยู่ก่อนหน้ามาให้

(CTRL U) จะเป็นคล้ายคำสั่งสำหรับ เปิด-ปิด การแก้ไขเรคอร์ด

(CTRL W) จะเก็บข้อมูลที่แก้ไขแล้วลงในแผ่นจานแม่เหล็กและจะกลับไปที่จุดเริ่มต้นของดีเบสทู (dBASE II) คือเครื่องหมาย

(CTRL Q) จะไม่รับการแก้ไขในเรคอร์ดที่กำลังแก้ไขนั้นและจะพิมพ์ตำแหน่งของเครื่องหมายเตรียมพร้อมรับคำสั่ง

จากนั้นกดคีย์ RETURN เพื่อกลับสู่จุดเริ่มต้นของดีเบส

หมายเหตุ (CTRL R) จะใช้สำหรับการแก้ไขเท่านั้น

การใช้คำสั่ง APPEND หากต้องการจะใส่ข้อมูลใหม่ ๆ เพิ่มเติมอีก ก็ทำได้โดยใช้คำสั่ง APPEND แล้ว<RET>ดีเบสทูจะแสดงออกทางจอภาพถึงเรคอร์ดถัดจากเรคอร์ดสุดท้ายของไฟล์นั้น ๆ มาให้ เช่น หากต้องการเพิ่มข้อมูลอีก 4 เรคอร์ดลงในไฟล์ WARRANTY ก็ใช้คำสั่ง APPEND จากนั้นบนจอภาพจะแสดงเรคอร์ดต่อมาในที่นี้คือ เรคอร์ดที่ 10 พร้อมทั้งเตรียมพร้อมที่จะรับข้อมูลลงในเรคอร์ดนี้ได้ทันที

ลองใส่เรคอร์ดต่อไปนี้ลงในไฟล์ WARRANTY

```
976121 998114 DU SAHAPANISH            89090.99 11/11/83
759110 764115 KB SAHAPANISH            31000.50 11/12/83
557333 569411 PT VIMONSILP LTD.        19500.50 11/12/83
546799 576888 PT BOON THAIYONT         19006.25 12/11/83
```

จากนั้นใช้คำสั่ง CTRL W เพื่อยุติการใส่ข้อมูลใหม่ และทำการเก็บข้อมูลใหม่ ๆ เหล่านี้ลงในจานแม่เหล็ก หรืออาจกด RET ในขณะที่ตัวเคอร์เซอร์อยู่ตำแหน่งแรกของเรคอร์ดใหม่ก็ได้เช่นกัน

หากใช้ CTRL Q ในการใส่ข้อมูลใหม่จะเป็นการลบเรคอร์ดนั้น ๆ ออก และกลับสู่จุดเริ่มต้นของดีเบสทู

การใช้คำสั่ง INSERT การแทรกเรคอร์ดใด ๆ จะสามารถวางไว้ที่ตำแหน่งใดก็ได้ในไฟล์ฐานข้อมูลเช่นการใช้ GO TO 5 จะเท่ากับเป็นการเรียกเรคอร์ดที่ 5 มาเป็นเรคอร์ดที่กำลังใช้อยู่ขณะนั้น และเมื่อ INSERT เรคอร์ดที่ 6 เดิมจะกลายเป็นเรคอร์ดที่ 7 และเรียงต่อไป

```
. GOTO 5
. INSERT
```

ในกรณีนี้เฉพาะหมายเลขประจำเรคอร์ดเท่านั้นที่แสดงออกมา การพิมพ์คำว่า INSERT เพียงอย่างเดียวจะเท่ากับเป็นการแทรกเรคอร์ดนั้นลงไปหลังจากเรคอร์ดที่กำลังปรากฏบนจอภาพ ตัวอย่างเช่นจะแทรกเรคอร์ดต่อไปนี้ให้เป็นเรคอร์ดที่ 6

```
978110 998998 KB SOOTSUWAN                30000.00 11/30/82
```

การแทรกเรคอร์ดใหม่ ๆ ลงในไฟล์ฐานข้อมูลด้วยคำสั่ง INSERT สามารถทำได้โดยใช้ 1 เรคอร์ดต่อ 1 คำสั่งเท่านั้น

หมายเหตุ

คำสั่ง *INSERT BEFORE* จะเป็นการแทรกเรคอร์ดที่ตำแหน่งก่อนหน้าเรคอร์ดที่กำลังแสดงอยู่เท่านั้น

คำสั่ง INSERT BLANK จะเป็นการแทรกเรคอร์ดที่ว่างเปล่า

การใช้คำสั่ง LIST, DISPLAY, SKIP SKIP GO/GO TO

เมื่อต้องการดูข้อมูลในไฟล์ที่ป้อนข้อมูลแล้วก็สามารถเรียกออกดูได้โดยตรงโดยใช้คำสั่งเกี่ยวกับการเรียกดู เช่น LIST, DISPLAY เป็นต้น การ LIST หรือ DISPLAY นี้อาจตั้งเงื่อนไขบางอย่างได้ด้วย เงื่อนไขดังกล่าวจึงมีประโยชน์มากสำหรับผู้ใช้โดยดีเบสทูจะเรียกออกมาเฉพาะเงื่อนไขที่เป็นจริงหรือ T เท่านั้น

เครื่องหมายที่ใช้แสดงความสัมพันธ์ (relational operation) ในการสร้างเงื่อนไขมี : ; ; >= ;< = และ = เครื่องหมายเหล่านี้เป็นตัวที่ใช้สร้างค่าของลอจิกขึ้น ซึ่งจะได้ผลเป็นจริง "T" หรือเท็จ "F" รายละเอียดการใช้เครื่องหมายเหล่านี้จะแสดงอยู่ในตอนท้าย

ในการทำงานที่เกี่ยวข้องกับเรคอร์ดต่าง ๆ แทนที่จะเกี่ยวข้องด้วยเป็นเรคอร์ดซึ่งไม่สะดวกเราก็สามารถตั้งสโคป (scope) หรือขอบข่ายการทำงานในดีเบสทูในแต่ละคำสั่งแทนที่จะให้ทำการคิดทำในเรคอร์ดเพียงอย่างเดียว หากใช้คำสั่ง SCOPE จะเป็นการคิดจำนวนของเรคอร์ดด้วย

รูปแบบการใช้คำสั่ง ในการใช้คือจะต้องใช้คำสั่งตามด้วยสโคปซึ่งมีด้วยกัน 3 กลุ่มดังนี้

(คำสั่ง) <ALL>

<RECORD n> (OFF) (FOR<เงื่อนไข>); เมื่อ n คือตัวเลขใด ๆ

<NEXT n>

ถ้าใช้ RECORD n จะทำตามคำสั่งเฉพาะเรคอร์ดที่ n เท่านั้น เช่น สั่ง LIST 5 ก็จะแสดงผลออกมาเฉพาะเรคอร์ดที่ 5 และหากเงื่อนไขไม่เป็นจริงก็จะไม่แสดงสิ่งใดออกมาแต่ถ้าใช้ LIST ALL ก็จะ LIST ข้อมูลทั้งหมดที่เงื่อนไขเป็นจริง

เงื่อนไข หมายถึงเงื่อนไขใด ๆ ที่กำหนดขึ้นมาโดยมีเครื่องหมายที่แสดงความสัมพันธ์ (relational operation) ดังที่แสดงไว้แล้ว

การใช้คำสั่ง LIST ในกรณีที่ต้องการจะดูรายละเอียดของแต่ละเรคอร์ดในไฟล์ฐานข้อมูลนั้นสามารถทำได้โดยใช้คำสั่ง LIST แต่ถ้าหากว่าในไฟล์ฐานข้อมูลนั้นมีจำนวนเรคอร์ดอยู่มากจนไม่สามารถแสดง

ให้ดูทั้งหมดได้บนจอภาพ ดีเบสทูจะทำการแสดงข้อมูลเรียงกันไปโดยภาพจะวิ่งและหยุดอยู่ที่เรคอร์ดล่างสุดของจอภาพและจะดันเรคอร์ดต่าง ๆ ขึ้นทีละบรรทัดขณะเดียวกันหากต้องการจะหยุดการแสดงนั้นชั่วขณะ เพื่อจะตรวจดูว่าถึงจุดที่สนใจหรือไม่ ก็สามารถทำได้โดยกด CTRL S

รูปแบบทั่ว ๆ ไปของคำสั่ง LIST คือ

LIST OFF SCOPE

หากใช้คำสั่ง LIST ที่ตามด้วย OFF, ดีเบสทูจะทำการแสดงผลโดยไม่แสดงหมายเลขประจำเรคอร์ด ส่วนรายละเอียดของสโคปเป็นไปตามที่กล่าวมาแล้ว

ลองพิมพ์คำสั่งต่อไปนี้ที่ใช้สโคปแบบต่าง ๆ กันแล้วสังเกตดูว่าผลที่แสดงออกมาเป็นอย่างไร

LIST OFF

LIST OFF ALL FOR MACHINTYPE = "DU"

LIST ALL FOR USRCOMPY = "BOON"

LIST OFF RECORD 3 FOR USRCOMPY = "SAHA"

```
. LIST OFF
954873 998717 DU BOON THAIYONT          90000.99 12/01/83
971981 997514 DU SOOTSUWAN              95000.00 11/10/83
750793 767231 KB P.NAKORN LTD.          30500.77 12/10/83
798717 765211 KB SIAM PANISH            31000.95 11/29/83
546798 567891 PT BOON THAIYONT          19005.95 12/10/83
978110 998998 KB SOOTSUWAN              30000.00 11/30/82
875179 768811 KB HOME ELECTRONIC        32000.91 10/05/83
557314 569811 PT HOME ELECTRONIC         7510.98 11/29/83
558111 568791 PT VIMONSILP LTD.          7810.98 11/29/83
951283 998111 DU VIMONSILP LTD.         91000.88 11/05/83
976121 998114 DU SAHAPANISH             89090.99 11/11/83
759110 764115 KB SAHAPANISH             31000.50 11/12/83
557333 569411 PT VIMONSILP LTD.         19500.50 11/12/83
546799 576888 PT BOON THAIYONT          19006.25 12/11/83
.
. LIST OFF ALL FOR MACHINTYPE="DU"
954873 998717 DU BOON THAIYONT          90000.99 12/01/83
971981 997514 DU SOOTSUWAN              95000.00 11/10/83
951283 998111 DU VIMONSILP LTD.         91000.88 11/05/83
976121 998114 DU SAHAPANISH             89090.99 11/11/83
.
. LIST ALL FOR USRCOMPY="BOON"
00001   954873 998717 DU BOON THAIYONT          90000.99 12/01/83
00005   546798 567891 PT BOON THAIYONT          19005.95 12/10/83
00014   546799 576888 PT BOON THAIYONT          19006.25 12/11/83
.
.
. LIST OFF RECORD 3 FOR USRCOMPY="SAHA"
.
.
. LIST OFF RECORD 3 FOR USRCOMPY="P.NAKORN"
```

```
750793 767231 KB P.NAKORN LTD.        30500.77 12/10/83
.
```

หมายเหตุ ไม่จำเป็นต้องพิมพ์ชื่อเต็มของบริษัทผู้ใช้ (user company) ในการค้นหาข้อมูลในไฟล์ เพราะการค้นหาจะทำการตรวจสอบ ถ้าตรงตามเงื่อนไขก็จะแสดงผล

LIST STRUCTURE	หมายถึง การสั่งให้แสดงโครงสร้างของไฟล์ฐานข้อมูลของไฟล์ที่ใช้อยู่ในขณะนั้น
LIST FILE	หมายถึง การสั่งให้แสดงไฟล์ทั้งหมด (file name) ทั้งหมดในฐานข้อมูลนั้น
LIST MEMORY	หมายถึง การสั่งให้แสดงตัวแปรหน่วยความจำที่ใช้ทั้งหมด (memory variables) ในขณะนั้น

ผลการทำงานด้วยดีเบสเป็นดังนี้

```
.
.
. LIST FILE

DATABASE FILES    # RCDS     LAST UPDATE
WARRANTY DBF      00014      12/12/83

. LIST STRUCTURE
STRUCTURE FOR FILE:  WARRANTY.DBF
NUMBER OF RECORDS:   00014
DATE OF LAST UPDATE: 12/12/83
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001      CARDNUM        C    006
002      SERIALNUM      C    006
003      MACHINTYPE     C    002
004      USRCOMPY       C    015
005      PURPRICE       N    010     002
006      DATE           C    008
** TOTAL **               00048
.
. LIST MEMORY
** TOTAL **      00 VARIABLES USED   00000 BYTES
USED
.
```

การใช้คำสั่ง DISPLAY รูปทั่วไปของการใช้คำสั่ง DISPLAY เป็นดังนี้

$$\text{DISPLAY} \begin{bmatrix} \text{ALL} \\ \text{RECORD n} \\ \text{NEXT n} \end{bmatrix} \quad \text{OFF} \quad \text{FOR} \quad \text{<เงื่อนไข>}$$

ข้อแตกต่างของคำสั่ง DISPLAY กับคำสั่ง LIST

1. ขอบเขตของคำสั่ง LIST ในขณะที่ไม่ตามด้วยสโคปใด หมายถึงทั้งหมด แต่สำหรับคำสั่ง DISPLAY ถ้าไม่ตามด้วยสโคปใดจะหมายถึงเฉพาะเรคอร์ดที่กำหนดเท่านั้น

2. คำสั่ง DISPLAY จะหยุดรอการกดคีย์ใดก็ได้หลังจากที่แสดงผลไปแล้ว 15 เรคอร์ด เพื่อเป็นการให้ผู้ใช้เครื่องในขณะนั้นทำการตรวจดูข้อมูลนั้นก่อนได้ แต่คำสั่ง LIST จะแสดงผลทั้งหมดโดยไม่มีการหยุดหากไม่ใช้ CTRL S

การใช้คำสั่ง GO/GOTO/SKIP ในกรณีที่ไฟล์ของระบบฐานข้อมูลนั้นมีข้อมูลจำนวนมาก และเราสนใจที่จะดูเพียงบางเรคอร์ดเท่านั้น หากระหว่างเรคอร์ดที่สนใจนั้นอยู่ห่างกันมากก็สามารถใช้คำสั่ง GO, GOTO และ SKIP ได้เพื่อจะดูเฉพาะเรคอร์ดที่สนใจเท่านั้น ซึ่งความหมายต่าง ๆ เป็นดังนี้

GO TOP	จะทำการแสดงผลของเรคอร์ดแรกในฐานข้อมูลนั้นบนจอภาพ
GO BOTTOM	จะทำการแสดงผลของเรคอร์ดสุดท้ายของฐานข้อมูล
GOTO n	จะแสดงผลของเรคอร์ดที่ n ที่ใส่ลงไปเท่านั้น
SKIP	จะทำการเลื่อนผลไป 1 เรคอร์ด ขึ้นบรรทัดใหม่ (ส่วนที่แสดงออกมาจะเป็นเฉพาะหมายเลขเรคอร์ดเท่านั้น)
SKIP +n/−n	จะทำการเลื่อนไปข้างหน้า หรือย้อนกลับไปเป็นจำนวน n เรคอร์ด

ลองใช้คำสั่งชุดต่อไปนี้แล้วสังเกตผลที่ได้จะทำให้เข้าใจมากยิ่งขึ้น

GO BOTTOM

DISPLAY

GO TOP

DISPLAY

```
. USE WARRANTY
. DISPLAY
00001  954873 998717 DU BOON THAIYONT    90000.99 12/01/83
. GO BOTTOM
. DISPLAY
00014  546799 576888 PT BOON THAIYONT    19006.25 12/11/83
.
.
. GO TOP
. DISPLAY
00001  954873 998717 DU BOON THAIYONT    90000.99 12/01/83
. SKIP
RECORD: 00002
. DISPLAY
00002  971981 997514 DU SOOTSUWAN        95000.00 11/10/83
. SKIP 7
RECORD: 00009
. DISPLAY
00009  558111 568791 PT VIMONSILP LTD.    7610.98 11/29/83
```

การใช้คำสั่ง DELETE, RECALL และ PACK

ในการที่จะลบเรคอร์ดใดทิ้งไป หลังจากที่ใช้คำสั่ง GOTO หรือ SKIP เพื่อที่จะหาเรคอร์ดที่ต้องลบแล้วให้ใส่คำสั่งว่า *"DELETE"* เข้าไป จะเท่ากับเป็นการลบเรคอร์ดนั้นทิ้งไป เช่น

GOTO 6

DISPLAY

DELETE

LIST

```
. DISPLAY
00006  978110 998998 KB SOOTSUWAN              30000.00 11/30/82
. DELETE
00001 DELETION(S)
. LIST
00001   954873 998717 DU BOON THAIYONT         90000.99 12/01/83
00002   971981 997514 DU SOOTSUWAN             95000.00 11/10/83
00003   750793 767231 KB P.NAKORN LTD.         30500.77 12/10/83
00004   798717 765211 KB SIAM PANISH           31000.95 11/29/83
00005   546798 567891 PT BOON THAIYONT         19005.95 12/10/83
00006  *978110 998998 KB SOOTSUWAN             30000.00 11/30/82
00007   875179 768811 KB HOME ELECTRONIC       32000.91 10/05/83
00008   557314 569811 PT HOME ELECTRONIC        7510.98 11/29/83
00009   558111 568791 PT VIMONSILP LTD.         7610.98 11/29/83
00010   951283 998111 DU VIMONSILP LTD.        91000.88 11/05/83
00011   976121 998114 DU SAHAPANISH            89090.99 11/11/83
00012   759110 764115 KB SAHAPANISH            31000.50 11/12/83
00013   557333 569411 PT VIMONSILP LTD.        19500.50 11/12/83
00014   546799 576888 PT BOON THAIYONT         19006.25 12/11/83
.
. DELETE ALL FOR USRCOMPY="SAHAPANISH"
00002 DELETION(S)
. LIST
00001   954873 998717 DU BOON THAIYONT         90000.99 12/01/83
00002   971981 997514 DU SOOTSUWAN             95000.00 11/10/83
00003   750793 767231 KB P.NAKORN LTD.         30500.77 12/10/83
00004   798717 765211 KB SIAM PANISH           31000.95 11/29/83
00005   546798 567891 PT BOON THAIYONT         19005.95 12/10/83
00006  *978110 998998 KB SOOTSUWAN             30000.00 11/30/82
00007   875179 768811 KB HOME ELECTRONIC       32000.91 10/05/83
00008   557314 569811 PT HOME ELECTRONIC        7510.98 11/29/83
00009   558111 568791 PT VIMONSILP LTD.         7610.98 11/29/83
00010   951283 998111 DU VIMONSILP LTD.        91000.88 11/05/83
00011  *976121 998114 DU SAHAPANISH            89090.99 11/11/83
00012  *759110 764115 KB SAHAPANISH            31000.50 11/12/83
00013   557333 569411 PT VIMONSILP LTD.        19500.50 11/12/83
00014   546799 576888 PT BOON THAIYONT         19006.25 12/11/83
. RECALL RECORD 6
00001 RECALL(S)
. LIST
00001   954873 998717 DU BOON THAIYONT         90000.99 12/01/83
00002   971981 997514 DU SOOTSUWAN             95000.00 11/10/83
```

```
00003  750793 767231 KB P.NAKORN LTD.          30500.77 12/10/83
00004  798717 765211 KB SIAM PANISH            31000.95 11/29/83
00005  546798 567891 PT BOON THAIYONT          19005.95 12/10/83
00006  978110 998998 KB SOOTSUWAN              30000.00 11/30/82
00007  875179 768811 KB HOME ELECTRONIC        32000.91 10/05/83
00008  557314 569811 PT HOME ELECTRONIC         7510.98 11/29/83
00009  558111 568791 PT VIMONSILP LTD.          7610.98 11/29/83
00010  951283 998111 DU VIMONSILP LTD.         91000.88 11/05/83
00011 *976121 998114 DU SAHAPANISH             89090.99 11/11/83
00012 *759110 764115 KB SAHAPANISH             31000.50 11/12/83
00013  557333 569411 PT VIMONSILP LTD.         19500.50 11/12/83
00014  546799 576888 PT BOON THAIYONT          19006.25 12/11/83
.
.
.
. DISPLAY FILE

DATABASE FILES    # RCDS     LAST UPDATE
WARRANTY DBF      00014      12/12/83
```

ในกรณีนี้หมายถึงการ GOTO 6 หรือไปยังเรคอร์ดที่ 6 แล้วแสดงผลเรคอร์ดที่ 6 เพื่อดูว่าต้องการลบแน่หรือไม่ ถ้าต้องการลบก็พิมพ์ DELETE จากนั้นลอง LIST ดูว่าเรคอร์ดที่ 6 ได้รับการลบหรือยัง

ในการสร้างเงื่อนไขสำหรับการลบนั้นสามารถทำได้โดยใช้ FOR แล้วตามด้วยเงื่อนไขเช่น

DELETE ALL FOR USRCOMPY = "SAHAPANISH"

คำสั่งนี้หมายถึงการลบทุกเรคอร์ดที่มีเงื่อนไข USRCOMPY = SAHAPANISH การลบทิ้งนั้นจะยังไม่เสร็จสมบูรณ์ในทันทีที่ใส่คำสั่งเสร็จ แต่จะมีเครื่องหมาย * อยู่ที่ RECORD นั้น ๆ ซึ่งจะยังคงเห็นเครื่องหมายนั้นในคำสั่ง LIST หรือ DISPLAY แต่ระบบดีเบสทูนี้จะไม่นำเรคอร์ดที่มีเครื่องหมาย * มาใช้ในขั้นตอนการทำงานเลย การยังคงไว้ก็เพื่อบางครั้งผู้ใช้อาจเรียกกลับมาใหม่ได้หรือให้ผู้ใช้ทราบว่าได้ทำการลบเรคอร์ดใดไปบ้าง

สำหรับกรณีจำเป็นที่จะใช้เรคอร์ดที่มีเครื่องหมาย * กลับมาใช้อีกสามารถทำได้โดยใช้คำสั่ง RECALL ดังนี้

RECALL<SCOPE>FOR<เงื่อนไข>เช่น RECALL RECORD 6

LIST

จะพบว่า RECORD 6 จะไม่มีเครื่องหมาย * อีกต่อไปและสามารถนำมาใช้งานได้อีกในระบบฐานข้อมูลต่อไป

หากใช้คำสั่ง DELETE แล้วและทำการตรวจสอบจนแน่ใจแล้วว่าต้องการที่จะลบเรคอร์ดนั้น ๆ ออกไปก็ทำได้โดยใช้คำสั่ง PACK ผลที่ได้คือ เรคอร์ดที่มีเครื่องหมาย * ทั้งหมดจะถูกลบออกไปและจะบอกอีกว่ายังมีเรคอร์ดอยู่ในไฟล์อีกเท่าใด การลบไฟล์ทั้งไฟล์ก็ทำได้โดยใช้คำสั่ง

DELETE FILE ชื่อฟิลด์นั้น

5

ตัวแปรและค่าคงที่ที่ใช้ในดีเบสทู

ในการทำงานบางชนิดจำเป็นจะต้องมีการใส่ค่าของตัวแปรใหม่ ๆ ที่อาจจะเป็นชื่อของสินค้าหรืออื่น ๆ และจำเป็นที่จะต้องมีการจำข้อมูลของตัวแปรใหม่เหล่านั้น ปัญหาเหล่านี้สามารถขจัดไปได้ด้วยการใช้คำสั่ง STORE เพราะจะเป็นการสั่งให้ดีเบสทูสร้างหน่วยความจำซึ่งเปลี่ยนค่าได้ (สูงสุด 64 ตัว) โดยจะใส่ค่าของอักขระหรือตัวเลขลงในตัวแปรหน่วยความจำที่ตั้งชื่อแล้วนั้น และชื่อของตัวแปรนั้นจะมีได้สูงสุดถึง 10 ตัวอักษร ต้องขึ้นต้นด้วยอักษร มีเครื่องหมายโคลอนได้ แต่ต้องไม่มีช่องว่าง (space)

รูปแบบของการใช้คำสั่ง STORE คือ

STORE<ตัวเลขหรือข้อความ>TO<หน่วยความจำที่แปรค่าได้>

เช่น

```
. STORE "MICROCOMPUTER RESEARCH LAB." TO NAME
. ? NAME
MICROCOMPUTER RESEARCH LAB.
.
. STORE 100.00 TO NUMBER
. ? NUMBER
    100.00
. GOTO 5
. DISPLAY
00005  546798 567891 PT BOON THAIYONT        19005.95 12/10/83
.
. STORE USRCOMPY TO CONAME
.
. ? CONAME
BOON THAIYONT
```

หากต้องการจะดูหน่วยความจำที่แปรค่าได้ทั้งหมดก็ใช้คำสั่ง DISPLAY MEMORY หรือ LIST MEMORY ก็ได้

เนื่องจากจำนวนหน่วยความจำที่แปรค่าได้มีจำกัดแค่ 64 ตัว บางครั้งจึงจำเป็นที่จะต้องลบของเดิมที่มีอยู่ออก สามารถทำได้โดยใช้คำสั่ง RELEASE เช่น RELEASE NAME ค่าของตัวแปรใน memory

name จะถูกลบออกไปและหากต้องหารลบค่าออกจากหน่วยความจำที่แปรค่าได้ทั้งหมดให้ใช้คำสั่ง RELEASE ALL

```
. DISPLAY MEMORY
NAME           (C)  MICROCOMPUTER RESEARCH LAB.
NUMBER         (N)   100.00
CONAME         (C)  BOON THAIYONT
** TOTAL **       03 VARIABLES USED  00048 BYTES USED
.
. RELEASE NAME
. LIST MEMORY
NUMBER         (N)   100.00
CONAME         (C)  BOON THAIYONT
** TOTAL **       02 VARIABLES USED  00021 BYTES USED
. RELEASE ALL
. LIST MEMORY
** TOTAL **       00 VARIABLES USED  00000 BYTES USED
.
```

การสั่งโดยตรงโดยใช้ "?"

การใช้คำสั่งนี้ ("?") เท่ากับเป็นการทำให้ใช้เครื่องไมโครคอมพิวเตอร์ในการคิดจำนวนเลขได้ทันทีแต่จะต้องตามหลังด้วยตัวกระทำทางคณิตศาสตร์

1. ( ) วงเล็บ
2. * เครื่องหมายคูณ
3. / เครื่องหมายหาร
4. + เครื่องหมายบวก
5. – เครื่องหมายลบ

ผลลัพธ์ที่ได้จะเป็นตัวเลขจำนวนจริงเท่านั้น เช่น

```
.
. ? 100/10
  10
. ? 10/3
  3
. ? 10/3.00
    3.33
. ? 10.000/3
  3.333
.
```

คำสั่งนี้จะแสดงผลลัพธ์ที่ได้จากการคำนวณ

ทดลองใส่คำสั่งต่อไปนี้

```
.
.
. GOTO 5
. ? USRCOMPY
BOON THAIYONT
```

ผลที่แสดงออกมาจะเป็นดังนี้ "BOON THAIYONT" ลองสังเกตดูว่าตัวอย่างของชุดของคำสั่งต่อไปนี้จะให้ผลออกมาเป็นอย่างไรบ้าง

```
.
. STORE USRCOMPY TO NAME
. STORE PURPRICE*0.01 TO CHARGE
. ? NAME
BOON THAIYONT
. ? CHARGE
   190.0595
.
```

หลังจากทดลองใส่ค่าตัวแปรต่าง ๆ ลงไปแล้ว หากต้องการจะลบค่าตัวแปรเหล่านี้ออกทั้งหมดให้ใช้คำสั่ง RELEASE ALL

การเลือกใช้ตัวกระทำที่แสดงความสัมพันธ์

ตัวกระทำที่แสดงความสัมพันธ์ทางลอจิกสามารถนำมาใช้ได้ในดีเบสทู โดยตัวกระทำเหล่านี้เขียนได้ในรูปแบบต่อไปนี้

( ) : วงเล็บสำหรับกลุ่มของข้อความ

.NOT.

.AND.

.OR.

.S. (ใช้สำหรับการหาค่าดูว่าในฟิลด์ที่กำหนดจะมีเรคอร์ดใดในฟิลด์นั้นที่ขึ้นต้นด้วยตัวอักษรที่ต้องการ)

ตัวกระทำที่แสดงความสัมพันธ์ระหว่างตัวแปรใด ๆ

< : น้อยกว่า	<> : ไม่เท่ากับ
> : มากกว่า	< = : น้อยกว่าหรือเท่ากับ
= : เท่ากับ	>=: มากกว่าหรือเท่ากับ

ลองพิมพ์คำสั่งต่อไปนี้แล้วสังเกตดูว่าผลที่ได้จะเป็นอย่างไร

```
. LIST FOR "THAI"$USRCOMPY
00001  954873 998717 DU BOON THAIYONT          90000.99 12/01/83
00005  546798 567891 PT BOON THAIYONT          19005.95 12/10/83
```

```
00014   546799 576888 PT BOON THAIYONT        19006.25 12/11/83
.
. LIST FOR CARDNUM="9" .OR. USRCOMPY="HOME ELECTRONIC"
00001   954873 998717 DU BOON THAIYONT        90000.99 12/01/83
00002   971981 997514 DU SOOTSUWAN            95000.00 11/10/83
00006   978110 998998 KB SOOTSUWAN            30000.00 11/30/82
00007   875179 768811 KB HOME ELECTRONIC      32000.91 10/05/83
00008   557314 569811 PT HOME ELECTRONIC       7510.98 11/29/83
00010   951283 998111 DU VIMONSILP.LTD.       91000.88 11/05/83
00011   976121 998114 DU SAHAPANISH           89090.99 11/11/83
.
. LIST FOR (CARDNUM="9" .AND. USRCOMPY="SOOT").AND.PURPRICE>90000
00002   971981 997514 DU SOOTSUWAN            95000.00 11/10/83
.

. LIST FOR (PURPRICE>10000.00 .AND. MACHINTYPE="DU")
00001   954873 998717 DU BOON THAIYONT        90000.99 12/01/83
00002   971981 997514 DU SOOTSUWAN            95000.00 11/10/83
00010   951283 998111 DU VIMONSILP LTD.       91000.88 11/05/83
00011   976121 998114 DU SAHAPANISH           89090.99 11/11/83
```

ตัวแปรของดีเบสทูใช้หลักการโดยการสร้างขึ้นในหน่วยความจำชนิดของตัวแปรจึงไม่เจาะจงถ้าหากว่ากำหนดให้ตัวแปรเก็บตัวเลขก็สามารถนำมาใช้ในทางการคำนวณได้ แต่ถ้าหากให้เก็บตัวอักษรก็จะใช้ได้ในทางลอจิก ตัวแปรจะมีประโยชน์ในการเขียนโปรแกรมดีเบสทูต่อไป

6

การแก้ไขโครงสร้างฐานข้อมูล

ในกรณีที่เราต้องการจะสร้างไฟล์ขึ้นมาใหม่โดยที่มีโครงสร้างของฟิลด์บางส่วนคล้ายกับของไฟล์ที่มีอยู่แต่เดิมแล้ว เช่น สมมติว่าต้องการสร้างไฟล์ขึ้นมาใหม่ให้ชื่อว่า USRFILE ซึ่งมีหลาย ๆ ฟิลด์ที่มีอยู่แล้วในไฟล์ WARRANTY คือต้องมีฟิลด์ CARDNUM, SERIALNUM และ USRCOMPY (ทั้ง 2 ไฟล์มีฟิลด์ที่มีชื่อเดียวกัน โครงสร้างเหมือนกันและข้อมูลทั้งชนิดและความยาวเท่ากัน) เราสามารถใช้คำสั่ง COPY เพื่อจะทำการลอกโครงสร้างจากไฟล์ WARRANTY ไปยังไฟล์ใหม่ที่ชื่อ USRFILE ได้ โดยใช้รูปแบบดังนี้

```
COPY TO USRFILE STRUCTURE FIELD CARDNUM,
SERIALNUM,USRCOMPY
```

หมายเหตุ การ COPY ไฟล์ในลักษณะนี้จะทำการ COPY เฉพาะฟิลด์ที่ระบุไว้เท่านั้น และจะลอกแบบเฉพาะโครงสร้างของแต่ละฟิลด์โดยไม่มีการ COPY ข้อมูลแต่ละเรคอร์ดในฟิลด์นั้นมาด้วย

ลองใช้คำสั่งต่อไปนี้เพื่อตรวจสอบดูว่าเกิดอะไรขึ้น โดยการคีย์คำสั่ง USE และ LIST ดังนี้

```
. USE USRFILE
. LIST STRUCTURE
STRUCTURE FOR FILE:   USRFILE.DBF
NUMBER OF RECORDS:    00000
DATE OF LAST UPDATE: 12/12/83
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001       CARDNUM       C     006
002       SERIALNUM     C     006
003       USRCOMPY      C     015
** TOTAL **                 00028
```

สิ่งที่ปรากฏคือ โครงสร้างของไฟล์ที่สร้างขึ้นมาใหม่โดยใช้การลอกเลียนมาจากของเดิม หากต้องการจะลอกเลียนทุก ๆ ฟิลด์ลงในไฟล์ใหม่ (ชื่อ TEMP) สามารถทำได้ด้วยคำสั่งรูปแบบดังนี้

```
. COPY TO TEMP STRUCTURE
. USE TEMP
. DISPLAY STRUCTURE
STRUCTURE FOR FILE:  TEMP.DBF
NUMBER OF RECORDS:   00000
DATE OF LAST UPDATE: 12/12/83
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001       CARDNUM      C     006
002       SERIALNUM    C     006
003       USRCOMPY     C     015
** TOTAL **                00028
```

(เนื่องจากขณะนี้เรากำลังใช้ไฟล์ USRFILE อยู่ดังนั้นทุก ๆ ฟิลด์ของ USRFILE จะถูก COPY ไปบนไฟล์ชื่อ TEMP)

เมื่อทดสอบดูแล้ว อาจจะลบไฟล์ TEMP ออกทั้งไฟล์ได้ในกรณีที่ไฟล์ TEMP ยังไม่ปิดต้องปิด file ก่อนโดยใช้ USE USRFILE (ต้องใช้ไฟล์อื่นก่อน เพราะไม่สามารถจะลบไฟล์ที่กำลังใช้อยู่ออกไปได้) หากไม่ปิดไฟล์ TEMP ก่อนผลจะเป็นดังนี้

```
.
. DELETE FILE TEMP
FILE IS CURRENTLY OPEN

.USE
.DELETE FILE TEMP
```

เมื่อกด RET จะพบว่า บนจอภาพจะแสดงผลดังนี้

```
FILE HAS BEEN DELETE
```

ซึ่งหมายถึงว่าไฟล์ได้ลบทิ้งแล้ว

รูปทั่วไปของคำสั่ง COPY

COPY TO ชื่อ FILE STRUCTURE ชื่อ ไฟล์ที่ต้องการลอกแบบ (การ COPY จะลอกเอาเฉพาะโครงสร้างเท่านั้น)

การเพิ่มฟิลด์ลงในฐานข้อมูลที่ยังว่างอยู่

เราสามารถเพิ่มหรือลบโครงสร้างของฟิลด์ได้โดยใช้คำสั่ง MODIFY การใช้คำสั่ง MODIFY จะเท่ากับเป็นการลบเอาข้อมูลที่มีอยู่ในฟิลด์นั้น ๆ ออกไปด้วย ดังนั้นจงอย่าใช้คำสั่ง MODIFY กับโครงสร้างของข้อมูลที่มีข้อมูลอยู่ด้วย

เนื่องจากไฟล์ชื่อ USRFILE DBF ยังไม่มีข้อมูลอยู่จึงสามารถใช้คำสั่ง MODIFY ได้ ดังตัวอย่าง

```
. USE USRFILE
. LIST STRUCTURE
STRUCTURE FOR FILE:  USRFILE.DBF
NUMBER OF RECORDS:   00000
DATE OF LAST UPDATE: 12/12/83
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001       CARDNUM     C     006
002       SERIALNUM   C     006
003       USRCOMPY    C     015
** TOTAL **               00028
. MODIFY STRUCTURE
```

ดีเบสทู จะตอบสนองทางจอภาพดังนี้

```
MODIFY ERASES ALL DATA RECORDS ... PROCEED? (Y/N) Y
```

เป็นการถามเพื่อเตือนให้ตรวจสอบดูอีกครั้งว่า ถ้าตอบ Y เท่ากับเป็นการลบข้อมูลของทุกเรคอร์ด ที่อยู่ในฟิลด์นั้นออกไป และดีเบสทูจะล้างจอภาพแล้วจะแสดงรายละเอียดของฟิลด์ทั้งหมด 16 ฟิลด์แรกในฐานข้อมูล จากนั้นใช้ลักษณะการแก้ไขทางจอภาพ (จากบทที่ 3) เพื่อทำการแทรกเครื่องหมาย ":" หลังคำว่า "CARD" และก่อนคำว่า "NUM" ดังนั้นชื่อของฟิลด์จะเปลี่ยนไปจาก "CARDNUM" เป็น "CARD : NUM"

หากต้องการจะใส่ฟิลด์เข้าไปใหม่อีก 3 ฟิลด์ชื่อว่า ADDRESS, PHONE และ STAFF NUM เข้าไปที่ฟิลด์ที่ 4 จากข้างต้นให้ใช้ CTRL X เพื่อเลื่อนเคอร์เซอร์ไปที่ฟิลด์ที่ 4 และใส่ฟิลด์ ลงไปตามรายละเอียดดังนี้

ADDRESS	C	30
PHONE	C	9
CONTACT	C	12
STAFFNUM	N	4

เมื่อต้องการออกจากคำสั่ง MODIFY สามารถเลือกทำได้ 2 ทางคือ

1. ใช้ CTRL W จะเป็นการเก็บเอาส่วนที่แก้ทั้งหมดลงในแผ่นจานแม่เหล็กแล้วจะกลับไปที่จุดเริ่มต้นของดีเบสทู

2. ใช้ CTRL Q เพื่อหยุดคำสั่ง MODIFY แล้วกลับไปที่จุดเริ่มต้น โดยไม่มีการบันทึกส่วนที่แก้ไขแล้วลงในแผ่นจานแม่เหล็กเลย แต่ถ้ากำลังทำการจัดโครงสร้างของข้อมูลโดยใช้คำสั่ง MODIFY อยู่นั้น จะสามารถออกจากคำสั่ง MODIFY นี้ได้โดยใช้ CTRL Q และจะไปจุดเริ่มต้นอีกเมื่อกด MODIFY STRUCTURE

ลองใช้คำสั่งต่อไปนี้ เพื่อตรวจดูว่าโครงสร้างใหม่นั้นเป็นไปตามที่ต้องการหรือไม่

```
. LIST STRUCTURE
STRUCTURE FOR FILE:  USRFILE.DBF
```

```
NUMBER OF RECORDS:    00000
DATE OF LAST UPDATE:  00/00/00
PRIMARY USE DATABASE
FLD        NAME        TYPE  WIDTH   DEC
001        CARD:NUM     C     006
002        SERIALNUM    C     006
003        USRCOMPY     C     015
004        ADDRESS      C     030
005        PHONE        C     007
006        CONTACT      C     012
007        STAFFNUM     N     004
** TOTAL **                 00081
```

การลบและแทรกฟิลด์

สมมติว่าในขณะนี้เราต้องการจะลบฟิลด์ CARD : NUM และ SERIAL ออก และต้องการจะแทรกฟิลด์ที่ชื่อ COUNTRY ลงไปหน้าฟิลด์ชื่อ PHONE โดยที่ COUNTRY นั้นเป็นชนิด C และมีความยาวให้เป็น 10

ตอนแรกนั้นใส่คำสั่ง MODIFY STRUCTURE และตอบคำถามด้วยตัว Y ตอนนี้ไฟล์ฐานข้อมูลจะยังคงว่างอยู่ ซึ่งเราสามารถจะทำการลบหรือเพิ่มฟิลด์ได้

บนจอภาพจะแสดงโครงสร้างของข้อมูลในไฟล์ USRFILE ซึ่งฟิลด์แรกก็คือ CARD : NUM ซึ่งเป็นฟิลด์ที่เราต้องการจะลบออกไป จากนั้นใช้ CTRL T เพื่อที่จะลบฟิลด์นี้ทิ้งไป คำสั่ง MODIFY จะทำการเลื่อนตำแหน่งของฟิลด์ที่อยู่ถัดออกไปขึ้นมาแทนที่ ใช้ CTRL T อีกครั้งเพื่อลบฟิลด์ SERIAL ออกไป ใช้ CTRL X เพื่อวางตำแหน่งของเคอร์เซอร์ที่ฟิลด์ชื่อ PHONE เมื่อใช้ CTRL N ดีเบสทูจะทำการเลื่อนฟิลด์ลงไป 1 ตำแหน่ง ทำให้มีที่ว่างที่จะแทรกฟิลด์ใหม่ลงไปได้ตรงตำแหน่งที่เคอร์เซอร์อยู่ในขณะนั้น จากนั้นใส่ชื่อฟิลด์ COUNTRY ลงไป กดแคร่สำหรับเลื่อนเพื่อเปลี่ยนตำแหน่งเคอร์เซอร์ไปที่ชนิดของข้อมูลแล้วระบุชนิด (C) ลงไป โดยทำนองเดียวกันจะสามารถใส่ความยาวของฟิลด์ (10 ตัวอักษร) ลงไปได้

ใช้ CTRL W เพื่อเก็บการแก้ไขโครงสร้างของข้อมูลในแผ่นจานแม่เหล็ก

สรุป คีย์คำสั่งในโหมด MODIFY

CTRL T	ทำการลบฟิลด์ที่ตำแหน่งของเคอร์เซอร์และจะเลื่อนฟิลด์อื่น ๆ ที่ติดกันขึ้นมา 1 ตำแหน่ง
CTRL Y	ทำการล้างฟิลด์ที่ตำแหน่งของเคอร์เซอร์ออกแต่จะไม่เลื่อนฟิลด์อื่น ๆ ขึ้นมา
CTRL N	จะเลื่อนฟิลด์ขณะนั้นลงไป 1 ตำแหน่งเพื่อจะได้มีที่ว่างสำหรับแทรกใหม่ลงไป

การเปลี่ยนโครงสร้างของข้อมูลในฟิลด์ที่ยังมีข้อมูลอยู่

ตามที่กล่าวมาแล้วคำสั่ง MODIFY STRUCTURE จะทำลายข้อมูลที่มีอยู่ในไฟล์ฐานข้อมูลนั้น แต่ในกรณีที่ต้องการจะทำการจัดโครงสร้างของข้อมูลที่ยังมีข้อมูลอยู่ ก็สามารถทำได้โดยใช้คำสั่ง COPY และ APPEND ควบคู่ไปกับคำสั่ง MODIFY STRUCTURE

ต่อไปนี้เราจะเพิ่มฟิลด์ชื่อ CHARGE เข้าไปในไฟล์ WARRANTY ซึ่งมีขั้นตอนในการทำงานดังนี้

1. COPY โครงสร้างของไฟล์ฐานข้อมูลที่เราต้องการไปเก็บ ไว้ที่ไฟล์ชั่วคราวไฟล์หนึ่ง (สร้างไฟล์นั้นขึ้นมาเอง)
2. ทำการแก้ไขส่วนต่าง ๆ ตามที่ต้องการลงในไฟล์ชั่วคราวอันนั้น
3. นำข้อมูลจากไฟล์แรกเริ่มที่มีอยู่แล้วมาเก็บไว้ในไฟล์ชั่วคราว
4. ตรวจสอบดูรายละเอียดของไฟล์ชั่วคราวนั้นด้วยคำสั่ง LIST เมื่อทุกเรคอร์ดถูกต้องดีแล้วก็ทำการ COPY กลับไปที่ไฟล์เริ่มต้น

ลองใช้ชุดของคำสั่งต่อไปนี้

```
.
. USE WARRANTY
. COPY TO TEMP STRUCTURE
. USE TEMP
. MODIFY STRUCTURE
MODIFY ERASES ALL DATA RECORDS ... PROCEED?
(Y/N) Y
```

MODIFY STRUCTURE (ตอบคำถามด้วยตัว 'Y')

ใช้คำสั่ง CTRL X เพื่อเลื่อนหาฟิลด์ที่ต้องการ จากนั้นใส่รายละเอียดของฟิลด์ใหม่ดังนี้

```
CHARGE   N   008   002

CTRL W
```

ใช้คำสั่ง LIST ดูว่าโครงสร้างที่สร้างมานั้นถูกต้องหรือไม่

```
.
.
. LIST STRUCTURE
STRUCTURE FOR FILE:   TEMP.DBF
NUMBER OF RECORDS:    00000
DATE OF LAST UPDATE:  12/12/83
PRIMARY USE DATABASE
FLD       NAME        TYPE WIDTH   DEC
001       CARDNUM      C    006
002       SERIALNUM    C    006
003       MACHINTYPE   C    002
004       USRCOMPY     C    015
005       PURPRICE     N    010    002
006       DATE         C    008
007       CHARGE       N    008    002
** TOTAL **                00056
```

จากนี้เราสามารถนำข้อมูลจากไฟล์ WARRANTY ไปไว้ในไฟล์ TEMP โดยใช้คำสั่งดังต่อไปนี้

```
.
. APPEND FROM WARRANTY
00014 RECORDS ADDED
.
```

หลังจากเสร็จสิ้นการใส่ข้อมูล สิ่งที่ปรากฏบนจอภาพจะบอกได้ว่า ได้ทำการสร้างเรคอร์ดใหม่ขึ้นมา 12 เรคอร์ดในไฟล์ TEMP และยังไม่มีข้อมูลใด ๆ เลยในฟิลด์ CHARGE ของทุก ๆ เรคอร์ดซึ่งจะตรวจสอบดูได้ด้วยคำสั่งต่อไปนี้

```
.
. LIST OFF ALL
954873 998717 DU BOON THAIYONT     90000.99 12/01/83  0.00
971981 997514 DU SOOTSUWAN         95000.00 11/10/83  0.00
750793 767231 KB P.NAKORN LTD.     30500.77 12/10/83  0.00
798717 765211 KB SIAM PANISH       31000.95 11/29/83  0.00
546798 567891 PT BOON THAIYONT     19005.95 12/10/83  0.00
978110 998998 KB SOOTSUWAN         30000.00 11/30/82  0.00
875179 768811 KB HOME ELECTRONIC   32000.91 10/05/83  0.00
557314 569811 PT HOME ELECTRONIC    7510.98 11/29/83  0.00
558111 568791 PT VIMONSILP LTD.     7610.98 11/29/83  0.00
951283 998111 DU VIMONSILP LTD.    91000.88 11/05/83  0.00
976121 998114 DU SAHAPANISH        89090.99 11/11/83  0.00
759110 764115 KB SAHAPANISH        31000.50 11/12/83  0.00
557333 569411 PT VIMONSILP LTD.    19500.50 11/12/83  0.00
546799 576888 PT BOON THAIYONT     19006.25 12/11/83  0.00
.
```

หากทุกอย่างถูกต้องเรียบร้อยแล้ว จึงทำการ COPY ไฟล์นี้กลับไปยังไฟล์เดิม (WARRANTY) โดยอาศัยชุดคำสั่งต่อไปนี้

```
.
.
. COPY TO WARRANTY
00014 RECORDS COPIED
.
. USE WARRANTY
. LIST OFF ALL
954873 998717 DU BOON THAIYONT     90000.99 12/01/83  0.00
971981 997514 DU SOOTSUWAN         95000.00 11/10/83  0.00
750793 767231 KB P.NAKORN LTD.     30500.77 12/10/83  0.00
798717 765211 KB SIAM PANISH       31000.95 11/29/83  0.00
546798 567891 PT BOON THAIYONT     19005.95 12/10/83  0.00
978110 998998 KB SOOTSUWAN         30000.00 11/30/82  0.00
875179 768811 KB HOME ELECTRONIC   32000.91 10/05/83  0.00
557314 569811 PT HOME ELECTRONIC    7510.98 11/29/83  0.00
558111 568791 PT VIMONSILP LTD.     7610.98 11/29/83  0.00
951283 998111 DU VIMONSILP LTD.    91000.88 11/05/83
```

```
976121 998114 DU SAHAPANISH              89090.99 11/11/83   0.00
759110 764115 KB SAHAPANISH              31000.50 11/12/83   0.00
557333 569411 PT VIMONSILP LTD.          19500.50 11/12/83   0.00
546799 576888 PT BOON THAIYONT           19006.25 12/11/83   0.00
```

การแก้ไขเปลี่ยนแปลงข้อมูลโดยทันทีด้วยคำสั่ง REPLACE และ CHANGE

การเปลี่ยนแปลงแก้ไขในเรคอร์ด บางเรคอร์ดหรือทุกเรคอร์ดสามารถทำได้อย่างรวดเร็ว โดยอาศัยคำสั่งต่อไปนี้

1. **คำสั่ง REPLACE** รูปแบบของคำสั่ง REPLACE เป็นดังนี้

REPLACE<SCOPE FIELD>WITH<DATA, FIELD>WITH<DATA..><FOR>ข้อความ

จะเห็นว่าคำสั่งมีความสำคัญมากพอควร คือมันจะทำการแทนค่าอะไรก็ได้ที่เรากำหนดขึ้นมาลงในฟิลด์ที่ตั้งชื่อให้ และสามารถจะทำการแทนค่าได้มากกว่าฟิลด์เดียวโดยใช้เครื่องหมายจุลภาค (,) หลังจากการแทนค่าครั้งแรกในส่วนที่ใส่ข้อความนั้นอาจจะเป็นรายละเอียดหรือคำสั่งคำนวณสำหรับหาค่าใด ๆ นอกจากนั้นเราอาจจะทำการแทนที่ได้โดยใช้คำสั่ง FOR โดยระบุเงื่อนไขที่แน่นอนให้ไป

หากค่าของราคาเครื่องจักร (PURPRICE) มีค่าน้อยกว่า 32000 จะต้องมีค่า CHARGE 2% ของราคาเครื่องจักร แต่ถ้าค่าของเครื่องจักรมากกว่า 32000 จะไม่ต้องเสียค่า CHARGE ในการซ่อม ดังนั้นในกรณีนี้ CHARGE = 0.0

ลองใช้คำสั่งต่อไปนี้ดู

```
 USE
.
. USE WARRANTY
. REPLACE ALL CHARGE WITH PURPRICE*0.02 FOR PURPRICE=10000
00000 REPLACEMENT(S)

.
. REPLACE ALL CHARGE WITH PURPRICE*0.02 FOR PURPRICE<32000
00009 REPLACEMENT(S)
. LIST OFF ALL
954873 998717 DU BOON THAIYONT      90000.99 12/01/83    0.00
971981 997514 DU SOOTSUWAN          95000.00 11/10/83    0.00
750793 767231 KB P.NAKORN LTD.      30500.77 12/10/83  610.01
798717 765211 KB SIAM PANISH        31000.95 11/29/83  620.01
546798 567891 PT BOON THAIYONT      19005.95 12/10/83  380.11
978110 998998 KB SOOTSUWAN          30000.00 11/30/82  600.00
875179 768811 KB HOME ELECTRONIC    32000.91 10/05/83    0.00
557314 569811 PT HOME ELECTRONIC     7510.98 11/29/83  150.21
558111 568791 PT VIMONSILP LTD.      7610.98 11/29/83  152.21
951283 998111 DU VIMONSILP LTD.     91000.88 11/05/83    0.00
976121 998114 DU SAHAPANISH         89090.99 11/11/83    0.00
759110 764115 KB SAHAPANISH         31000.50 11/12/83  620.01
557333 569411 PT VIMONSILP LTD.     19500.50 11/12/83  390.01
```

```
546799 576888 PT BOON THAIYONT    19006.25 12/11/83  380.12
```

คำสั่ง CHANGE รูปแบบของคำสั่ง CHANGE เป็นดังนี้

CHANGE <scope> FIELD<list> FOR<expression>

ในกรณีนี้อย่างน้อยที่สุดจะต้องมีชื่อของฟิลด์ 1 ฟิลด์ แต่ชื่อของฟิลด์หลาย ๆ ฟิลด์จะ List ออกมาหากแยกด้วยเครื่องหมายจุลภาค (,)

คำสั่งนี้เมื่อสั่งทำงานในตอนแรกจะทำการเรคอร์ดตามเงื่อนไข จากนั้นจะทำการแสดงชื่อของเรคอร์ดรายละเอียดและจะหยุดถาม

ลองใช้คำสั่งต่อไปนี้

```
CHANGE RECORD 3 FIELD MACHINTYPE,DATE
```

ผลที่แสดงออกมาจะเป็นดังนี้

```
RECORD: 00003

MACHINTYPE: KB
CHANGE?
```

กด KEY RET เพื่อไม่ให้มีการเปลี่ยนแปลงในฟิลด์นี้ จากนั้นดีเบสทูจะทำการแสดงผลของฟิลด์หนึ่งที่ปรากฏในคำสั่ง CHANGE ดังนี้

```
DATE: 12/10/83
CHANGE? 12
TO        11
```

หากเราต้องการจะเปลี่ยนเดือน จาก 12 เป็น 11 ทำได้โดยใส่

12

ดีเบสทูจะถามว่า

TO ?

ใส่ 11 RET

ดีเบสทูจะแสดงผลที่เปลี่ยนแปลงแล้วดังนี้

```
DATE: 11/10/83
CHANGE?
```

และจะถามอีกว่า

CHANGE ?

หากไม่ต้องการจะแก้ไขส่วนใดอีกก็กด RET

หากต้องการจะหยุดการแก้ไขนี้ที่จุดใดจุดหนึ่งก็เพียงแต่กด ESC

หมายเหตุ ในกรณีที่ฟิลด์นั้นไม่มีข้อมูลอยู่เลย และเราต้องการจะใส่ข้อมูลเข้าไปในฟิลด์นั้น ก็เพียงแต่คีย์ space เข้าไป

รูปที่สมบูรณ์ของคำสั่ง CHANGE เป็นดังนี้

CHANGE <scope> FIELD <list> FOR <expression>

7

การจัดเรียงลำดับในฐานข้อมูล

ข้อมูลที่เก็บไว้ในฐานข้อมูลนั้นโดยทั่วไปแล้วมักจะใส่เข้าไปโดยไม่มีการจัดเรียงมาก่อน ดังนั้น เมื่อต้องการจะจัดข้อมูลให้เป็นระเบียบสามารถทำได้โดยอาศัยคำสั่ง SORT และ INDEX

การใช้คำสั่ง SORT

คำสั่ง SORT นี้จะทำการจัดลำดับข้อมูลตามลำดับในฟิลด์ใด ๆ ก็ตามที่เราระบุไป ซึ่งฟิลด์ที่ระบุเรียกว่าคีย์ และคำสั่งนี้เมื่อสั่งงานแล้วจะได้ไฟล์ใหม่ที่แตกต่างจากไฟล์แรกเริ่มก่อนจะทำการ SORT และสามารถจะจัดลำดับการเรียงฟิลด์ที่ต้องการได้ว่าจะให้เรียงลำดับจากมากไปน้อยหรือจากน้อยไปมาก (ASCENDING or DESCENDING) สมมติว่าเราทำการจัดลำดับในฟิลด์ CARDNUM สิ่งที่ต้องทำคือ

```
.
.
. USE WARRANTY
. SORT ON CARDNUM TO SORTFILE
SORT COMPLETE
.
. USE SORTFILE
. LIST ALL OFF
546798 567891 PT BOON THAIYONT          19005.95 12/10/83    380.11
546799 576888 PT BOON THAIYONT          19006.25 12/11/83    380.12
557314 569811 PT HOME ELECTRONIC         7510.98 11/29/83    150.21
557333 569411 PT VIMONSILP LTD.         19500.50 11/12/83    390.01
558111 568791 PT VIMONSILP LTD.          7610.98 11/29/83    152.21
750793 767231 KB P.NAKORN LTD.          30500.77 11/10/83    610.01
759110 764115 KB SAHAPANISH             31000.50 11/12/83    620.01
798717 765211 KB SIAM PANISH            31000.95 11/29/83    620.01
875179 768811 KB HOME ELECTRONIC        32000.91 10/05/83      0.00
951283 998111 DU VIMONSILP LTD.         91000.88 11/05/83      0.00
954873 998717 DU BOON THAIYONT          90000.99 12/01/83      0.00
971981 997514 DU SOOTSUWAN              95000.00 11/10/83      0.00
976121 998114 DU SAHAPANISH             89090.99 11/11/83    600.00
978110 998998 KB SOOTSUWAN              30000.00 11/30/82
.
```

จะเห็นได้จากผลลัพธ์ที่แสดงว่าฟิลด์ CARDNUM ได้รับการจัดเรียงอย่างเป็นระเบียบเรียบร้อยแล้วตามตัวอักษรจากน้อยไปหามาก

หากไม่ระบุลงไปว่าต้องการจัดลำดับในลักษณะใดแล้วดีเบสทูจะทำการจัดลำดับจากน้อยไปมากให้ และหลังจากทำการจัดเรียงเสร็จแล้ว ลักษณะบนจอภาพจะเป็นดังนี้

SORT COMPLETE

เมื่อต้องการจะดูว่าฟิลด์ CARDNUM นั้นมีการจัดเรียงอย่างที่สั่งหรือไม่ จะต้องตรวจดูในไฟล์ SORTFILE ดังนี้

USE SORTFILE

LIST ALL OFF

รูปเต็มของคำสั่ง SORT เป็นดังนี้

SORT ON <ชื่อ field> TO <ชื่อ file ใหม่> <DESCENDING>

ส่วนของชื่อฟิลด์นั้นจะบอกว่าต้องทำการจัดเรียงในฟิลด์ใด โดยการจัดเรียงนั้นอาจจะเป็นได้ทั้งข้อมูลที่เป็นตัวอักษรและตัวเลข แต่ต้องไม่ใช่ข้อมูลทางลอจิก

เมื่อทำการ SORT เสร็จแล้วผลที่ได้จะเก็บไว้ในไฟล์หนึ่ง (ในที่นี้คือ SORTFILE) หากต้องการนำผลนั้นมาเก็บไว้ในไฟล์ชื่อ WARRANTY อีก ก็ทำได้โดยอาศัยการ COPY ดังนี้

```
.
. USE SORTFILE
. COPY TO WARRANTY
00014 RECORDS COPIED
. USE WARRANTY
. LIST OFF
546798 567891 PT BOON THAIYONT         19005.95 12/10/83     380.11
546799 576888 PT BOON THAIYONT         19006.25 12/11/83     380.12
557314 569811 PT HOME ELECTRONIC        7510.98 11/29/83     150.21
557333 569411 PT VIMONSILP LTD.        19500.50 11/12/83     390.01
558111 568791 PT VIMONSILP LTD.         7610.98 11/29/83     152.21
750793 767231 KB P.NAKORN LTD.         30500.77 11/10/83     610.01
759110 764115 KB SAHAPANISH            31000.50 11/12/83     620.01
798717 765211 KB SIAM PANISH           31000.95 11/29/83     620.01
875179 768811 KB HOME ELECTRONIC       32000.91 10/05/83       0.00
951283 998111 DU VIMONSILP LTD.        91000.88 11/05/83       0.00
954873 998717 DU BOON THAIYONT         90000.99 12/01/83       0.00
971981 997514 DU SOOTSUWAN             95000.00 11/10/83       0.00
976121 998114 DU SAHAPANISH            89090.99 11/11/83       0.00
978110 998998 KB SOOTSUWAN             30000.00 11/30/82     600.00
```

ระวัง: เมื่อสั่ง SORT ห้ามใส่ชื่อไฟล์เป็นไฟล์เดียวกับไฟล์ของฟิลด์ที่ทำการ SORT เพราะการ SORT จะไปทำการสร้างไฟล์ทับไฟล์เดิมและผลที่เกิดคือจะทำการจัดเรียงไม่ได้

ข้อจำกัดในการใช้คำสั่ง SORT

1. จะทำการ SORT ฟิลด์ 2 ฟิลด์ แล้วเก็บในไฟล์เดียวกันไม่ได้

2. เมื่อทำการ SORT เสร็จแล้ว เมื่อใส่ข้อมูลเข้าไปใหม่ ข้อมูลนั้นจะไม่ไปอยู่ที่ตำแหน่งตามการ SORT แต่จะอยู่ต่อท้ายเรคอร์ดสุดท้ายไปเรื่อย ๆ หากจะทำการจัดเรียงลำดับใหม่ต้องสั่ง SORT อีก

การใช้คำสั่ง INDEX

การใช้คำสั่ง INDEX จะใช้ในกรณีที่ต้องการจัดเรียงลำดับของข้อมูลที่ต้องการโดยมีการกำหนดคีย์ ซึ่ง *คีย์* หมายถึง ฟิลด์ของฐานข้อมูลหรือฟิลด์ที่ต้องการให้ทำการจัดลำดับด้วยคำสั่ง INDEX ในฐานข้อมูลจะทำการจัดลำดับเฉพาะในคีย์ที่สนใจเท่านั้น

รูปแบบของคำสั่งทั่วไปของคำสั่ง INDEX

INDEX ON <key (variable/expression)> TO <index filename>

จากรูปแบบนี้จะทำการสร้างไฟล์ขึ้นมาใหม่ซึ่งไฟล์นี้จะมีส่วนอธิบายรายละเอียดเป็น .NDX และเฉพาะข้อมูลที่อยู่ในคีย์เท่านั้นที่จะทำการจัดลำดับ ถึงแม้จะดูเหมือนว่าข้อมูลทั้งหมดจะถูกจัดลำดับก็ตาม

สมมติว่า เราต้องการใช้คำสั่ง INDEX กับฟิลด์ USRCOMPY ด้วยชุดคำสั่งต่อไปนี้

USE WARRANTY

INDEX ON USRCOMPY TO USRIN

```
. USE WARRANTY
. INDEX ON USRCOMPY TO USRIN
00014 RECORDS INDEXED
.
.
. USE WARRANTY INDEX USRIN
. LIST
00001  546798 567891 PT BOON THAIYONT       19005.95 12/10/83   380.11
00002  546799 576888 PT BOON THAIYONT       19006.25 12/11/83   380.12
00011  954873 998717 DU BOON THAIYONT       90000.99 12/01/83     0.00
00003  557314 569811 PT HOME ELECTRONIC      7510.98 11/29/83   150.21
00009  875179 768811 KB HOME ELECTRONIC     32000.91 10/05/83     0.00
00006  750793 767231 KB P.NAKORN LTD.       30500.77 11/10/83   610.01
00007  759110 764115 KB SAHAPANISH          31000.50 11/12/83   620.01
00013  976121 998114 DU SAHAPANISH          89090.99 11/11/83     0.00
00008  798717 765211 KB SIAM PANISH         31000.95 11/29/83   620.01
00012  971981 997514 DU SOOTSUWAN           95000.00 11/10/83     0.00
00014  978110 998998 KB SOOTSUWAN           30000.00 11/30/82   600.00
00004  557333 569411 PT VIMONSILP LTD.      19500.50 11/12/83   390.01
00005  558111 568791 PT VIMONSILP LTD.       7610.98 11/29/83   152.21
00010  951283 998111 DU VIMONSILP LTD.      91000.88 11/05/83     0.00
. GO BOTTOM
. DISPLAY
00010  951283 998111 DU VIMONSILP LTD.      91000.88 11/05/83     0.00
.
. GO TOP
. DISPLAY
00001  546798 567891 PT BOON THAIYONT       19005.95 12/10/83   380.11
. SKIP
RECORD: 00002
.
. DISPLAY
00002  546799 576888 PT BOON THAIYONT       19006.25 12/11/83   380.12
.
```

```
. SKIP
RECORD: 00011
. DISPLAY
00011  954873 998717 DU BOON THAIYONT        90000.99 12/01/83        0.00
.
```

ระวัง: เมื่อใช้คำสั่ง INDEX ต้องระวังอย่าให้ชื่อไฟล์ที่ใช้เก็บการ INDEX เป็นไฟล์เดียวกับไฟล์เริ่มต้น ผลของการกระทำการอินเด็กซ์ตามคำสั่งดังกล่าวเป็นดังนี้

หมายเหตุ

เราอาจจะใช้คำสั่งนี้ในกรณีที่ต้องการจัดลำดับที่ซ้อนกันได้ ดังนี้

INDEX ON FIELD 1 + FIELD 2 + FIELD 3

เมื่อต้องการความรวดเร็วในการสร้าง INDEX ไฟล์เราจำเป็นต้องกำหนดค่าที่แน่นอนลงในส่วนของคำสั่ง USE ตามตัวอย่าง

USE WARRANTY INDEX USRIN

เมื่อใช้คำสั่งนี้จะพบว่าไฟล์ที่ใช้งานคือ USRIN .NDX ไม่ใช่ไฟล์ WARRANTY

หมายเหตุ คำสั่งที่ใช้ในการกำหนดตำแหน่ง เช่น GO, GO BOTTOM, SKIP และอื่น ๆ กับคำสั่ง INDEX จะมีผลกับลำดับที่ของการจัดเรียงลำดับมากกว่าที่จะเป็นตามแบบไฟล์ในระบบฐานข้อมูลอื่น ๆ

ตัวอย่างเช่น คำสั่ง GO BOTTOM จะนำไปสู่เรคอร์ดที่เป็นลำดับสุดท้ายในการจัดลำดับแทนที่จะเป็นเรคอร์ดลำดับสุดท้ายของไฟล์นั้น จากนั้นลองพิมพ์

LIST ALL OFF แล้วสังเกตดูว่าลำดับของแต่ละเรคอร์ดเปลี่ยนไปอย่างไรต่อไป

ลองใช้คำสั่งต่อไปนี้

GO BOTTOM

DISPLAY

จะสังเกตดูว่าเรคอร์ดสุดท้ายจะเป็น "VIMONSILP LTD." แทนที่จะเป็น RECORD สุดท้ายของไฟล์นั้น

นอกจากนี้เรายังสามารถที่จะกำหนดฟิลด์ที่แตกต่างกันด้วยตัวเลขต่าง ๆ ในกรณีที่ต้องการทำงานที่แตกต่างกันได้

การค้นหาข้อมูลในไฟล์ชนิดอินเด็กซ์ ในการใช้คำสั่ง INDEX จะสามารถหาตำแหน่งของเรคอร์ดที่ต้องการได้อย่างรวดเร็ว โดยการกำหนดตำแหน่งที่แน่นอนโดยใช้คำสั่ง FIND.

การใช้คำสั่ง FIND

ข้อมูลในฐานข้อมูลจะได้รับการค้นหาโดยใช้คำสั่ง FIND ได้ต่อเมื่อได้ทำการ INDEX ข้อมูลเหล่านั้นก่อน รูปทั่วไปของคำสั่ง FIND เป็นดังนี้

FIND <character string>

คำสั่งนี้จะสั่งให้ดีเบสทูหาเรคอร์ดแรกในระบบฐานข้อมูลชนิดอินเด็กซ์ไฟล์ซึ่งในคีย์ที่ใช้เช่นเดียวกับ character string โดยทั่วไปแล้วคำสั่ง FIND จะใช้เวลาประมาณ 2 วินาทีในการหาข้อมูลที่ต้องการจากจานแม่เหล็ก และ character string จะมีความยาวเท่าใดก็ได้ตามแต่ต้องการ

หมายเหตุ คำสั่ง FIND จะหาเฉพาะเรคอร์ดแรกในคีย์ที่เหมือนกับค่าใน character string เท่านั้น

ลองใส่คำสั่งต่อไปนี้

USE WARRANTY INDEX USRIN

FIND HOME ELECTRONIC

DISPLAY NEXT 3

FIND SIAM

DISPLAY NEXT 3

FIND VIMONSILP

DISPLAY NEXT 1

ในกรณีที่ไม่มีเรคอร์ดใดเลยที่เหมือนกับค่าที่เราต้องการดีเบสทูจะแสดงผลออกมาดังนี้

NO FIND

ลองใส่คำสั่ง

FIND CHARLES

ความหมายของ NO FIND คือไม่มีเรคอร์ดใดใน INDEX (นั่นคือ USRCOMPY) เรียกว่า CHARLES เลย ลองพิจารณาจากการทำงานตามตัวอย่างที่กล่าวแล้วดังนี้

```
.
. USE WARRANTY INDEX USRIN
. FIND HOME ELECTRONIC
. DISPLAY NEXT 3
00003   557314 569811 PT HOME ELECTRONIC     7510.98 11/29/83    150.21
00009   875179 768811 KB HOME ELECTRONIC    32000.91 10/05/83      0.00
00006   750793 767231 KB P.NAKORN LTD.      30500.77 11/10/83    610.01
.
. FIND SIAM
. DISPLAY NEXT 3
00008   798717 765211 KB SIAM PANISH        31000.95 11/29/83    620.01
00012   971981 997514 DU SOOTSUWAN          95000.00 11/10/83      0.00
00014   978110 998998 KB SOOTSUWAN          30000.00 11/30/82    600.00
.
. FIND VIMONSILP
. DISPLAY NEXT 1
00004   557333 569411 PT VIMONSILP LTD.     19500.50 11/12/83    390.01
.
. FIND THONGCHAI
NO FIND
.
. FIND CHARLES
NO FIND
.
```

การใช้คำสั่ง LOCATE

ในกรณีที่ต้องการทราบข้อมูลเฉพาะที่อยู่ในไฟล์ซึ่งไม่ได้อินเด็กซ์ และกำลังใช้อยู่นั้นสามารถหาได้โดยใช้คำสั่ง LOCATE

ในขณะนี้เรากำลังใช้ USRIN.NDX เป็นฐานข้อมูลอยู่และคีย์ที่อินเด็กซ์ก็คือ USRCOMPY และเราก็ต้องการหารายละเอียดของข้อมูลโดยใช้ MACHINTYPE เป็นคีย์จะเห็นว่าเราไม่สามารถใช้คำสั่ง FIND ได้เนื่องจาก MACHINTYPE เป็นคีย์ที่ไม่ได้อินเด็กซ์เราจึงใช้คำสั่ง LOCATE แทน

รูปทั่วไปของคำสั่ง LOCATE เป็นดังนี้

LOCATE <scope> FOR <expression>

คำสั่ง CONTINUE

สมมติว่าข้อมูลที่เราต้องการจะหามานั้น คือข้อมูลในเรคอร์ดใด ๆ ที่ในฟิลด์ MACHINTYPE เป็น 'DU' จะหาได้โดยใช้คำสั่งต่อไปนี้ (file ที่ใช้จะต้องเป็น USRIN)

LOCATE ALL FOR MACHINTYPE = 'DU'

ดีเบสทูจะแสดง :	RECORD 00011
ให้พิมพ์ว่า	DISPLAY
	CONTINUE
ดีเบสทูจะแสดง :	RECORD 00013
พิมพ์ว่า	DISPLAY
	CONTINUE
ดีเบสทูจะแสดง :	RECORD 00012
พิมพ์ว่า	DISPLAY
	CONTINUE
ดีเบสทูจะแสดง :	RECORD 00010
พิมพ์ว่า	DISPLAY
	CONTINUE
ดีเบสทูจะแสดงผลออกมาดังนี้ :	END OF FILE ENCOUNTERED

ผลของการทำงานดังกล่าวเป็นดังนี้

```
.
. USE WARRANTY INDEX USRIN
. LOCATE ALL FOR MACHINTYPE="DU"
RECORD: 00011
. DISPLAY
00011  954873 998717 DU BOON THAIYONT       90000.99 12/01/83       0.00
.
. CONTINUE
RECORD: 00013
. DISPLAY
00013  976121 998114 DU SAHAPANISH          89090.99 11/11/83       0.00
```

```
.
. CONTINUE
RECORD: 00012
. DISPLAY
00012  971981 997514 DU SOOTSUWAN            95000.00 11/10/83
.                                                               0.00
. CONTINUE
RECORD: 00010
. DISPLAY
00010  951283 998111 DU VIMONSILP LTD.       91000.88 11/05/83
.                                                               0.00
. CONTINUE
END OF FILE ENCOUNTERED
```

การเพิ่มหรือลบเรคอร์ดออกจากฐานข้อมูลที่อินเด็กซ์

โดยทั่วไปแล้วคำสั่ง INDEX มักจะใช้เพียงไม่กี่ครั้ง (โดยมากครั้งเดียว) ในไฟล์ที่กำหนดให้ หากต้องการใส่ข้อมูลใหม่ ๆ เข้าไปอีกก็ใช้คำสั่ง APPEND ซึ่งข้อมูลที่ใส่เข้าไปใหม่นี้จะจัดตัวเองให้เข้ากับคำสั่ง INDEX (ที่สั่งไว้เดิม) โดยอัตโนมัติ ลองสังเกตผลที่ปรากฏจากชุดคำสั่งต่อไปนี้

USE WARRANTY INDEX USRIN

APPEND

จากนั้นให้ใส่เรคอร์ดใหม่เข้าไปดังนี้

985731   998432 PT. PATANAPAN   5000.99   11/12/83

0.00

เมื่อใส่เรคอร์ดนี้หมดแล้วใช้ CTRL W (เพื่อยุติคำสั่ง APPEND)

ให้พิมพ์ GO TOP

LIST

จะเห็นว่าเรคอร์ดที่ใส่เข้าไปใหม่นี้แทนที่จะอยู่ที่ตำแหน่งสุดท้ายของไฟล์กลับถูกนำมาแทรกอยู่ในกลุ่มของเรคอร์ดที่ทำการ INDEX ฟิลด์ USRCOMPY ไว้ก่อนแล้ว จากนั้นใช้คำสั่ง

USE WARRANTY

LIST ALL

จะพบว่าเรคอร์ดที่ใส่เข้าไปใหม่ในไฟล์ USRIN.NDX นั้นถูกนำมาใส่ไว้ในเรคอร์ดสุดท้ายของไฟล์ WARRANTY ด้วย ในทำนองเดียวกันการลบเรคอร์ดใดออกจากไฟล์ที่ทำการ INDEX แล้วก็จะถูกลบออกจากไฟล์ที่เป็นไฟล์เริ่มต้นของไฟล์ indexed ด้วย

สรุป การเปลี่ยนแปลงใด ๆ ที่ทำกับคีย์ฟิลด์ด้วยคำสั่ง APPEND, EDIT, REPLACE หรือ PACK ในฐานข้อมูลจะมีผลถึงอินเด็กซ์ไฟล์ที่ใช้อยู่ด้วย อย่างไรก็ตามอินเด็กซ์ไฟล์อื่น ๆ ที่ทำการอินเด็กซ์ข้อมูลชุดเดียวกันจะไม่ถูกแก้ไขไปด้วย ส่วนเหล่านี้จะถูก *update* เหมือนกับว่าแต่ละเรคอร์ดถูกเพิ่มเข้าไปโดยใช้คำสั่ง INDEX และ REINDEX ใหม่อีกครั้งจะเห็นได้ว่าสิ่งเหล่านี้เป็นสิ่งเดียวที่คำสั่งการแก้ไขไม่สามารถครอบคลุมไปได้

ผลของการกระทำดังกล่าวแสดงให้เห็นผลลัพธ์ดังนี้

```
. USE WARRANTY INDEX USRIN
. APPEND
RECORD # 00015
. LIST
00001  546798 567891 PT BOON THAIYONT        19005.95 12/10/83    380.11
00002  546799 576888 PT BOON THAIYONT        19006.25 12/11/83    380.12
00011  954873 998717 DU BOON THAIYONT        90000.99 12/01/83      0.00
00003  557314 569811 PT HOME ELECTRONIC       7510.98 11/29/83    150.21
00009  875179 768811 KB HOME ELECTRONIC      32000.91 10/05/83      0.00
00006  750793 767231 KB P.NAKORN LTD.        30500.77 11/10/83    610.01
00015  985731 998432 PT PATANAPAN             5000.99 11/12/83      0.00
00007  759110 764115 KB SAHAPANISH           31000.50 11/12/83    620.01
00013  976121 998114 DU SAHAPANISH           89090.99 11/11/83      0.00
00008  798717 765211 KB SIAM PANISH          31000.95 11/29/83    620.01
00012  971981 997514 DU SOOTSUWAN            95000.00 11/10/83      0.00
00014  978110 998998 KB SOOTSUWAN            30000.00 11/30/82    600.00
00004  557333 569411 PT VIMONSILP LTD.       19500.50 11/12/83    390.01
00005  558111 568791 PT VIMONSILP LTD.        7610.98 11/29/83    152.21
00010  951283 998111 DU VIMONSILP LTD.       91000.88 11/05/83      0.00
.
.
. USE WARRANTY
. LIST ALL
00001  546798 567891 PT BOON THAIYONT        19005.95 12/10/83    380.11
00002  546799 576888 PT BOON THAIYONT        19006.25 12/11/83    380.12
00003  557314 569811 PT HOME ELECTRONIC       7510.98 11/29/83    150.21
00004  557333 569411 PT VIMONSILP LTD.       19500.50 11/12/83    390.01
00005  558111 568791 PT VIMONSILP LTD.        7610.98 11/29/83    152.21
00006  750793 767231 KB P.NAKORN LTD.        30500.77 11/10/83    610.01
00007  759110 764115 KB SAHAPANISH           31000.50 11/12/83    620.01
00008  798717 765211 KB SIAM PANISH          31000.95 11/29/83    620.01
00009  875179 768811 KB HOME ELECTRONIC      32000.91 10/05/83      0.00
00010  951283 998111 DU VIMONSILP LTD.       91000.88 11/05/83      0.00
00011  954873 998717 DU BOON THAIYONT        90000.99 12/01/83      0.00
00012  971981 997514 DU SOOTSUWAN            95000.00 11/10/83      0.00
00013  976121 998114 DU SAHAPANISH           89090.99 11/11/83      0.00
00014  978110 998998 KB SOOTSUWAN            30000.00 11/30/82    600.00
00015  985731 998432 PT PATANAPAN             5000.99 11/12/83      0.00
```

การใช้คำสั่ง UPDATE

คำสั่ง UPDATE จะเป็นการนำข้อมูลจากฐานข้อมูลของไฟล์ที่ระบุ (ไฟล์ที่ไม่ได้ใช้ (USE) ขณะนั้น) เพื่อนำมาใช้ในไฟล์ที่กำลังใช้อยู่

รูปทั่วไปของคำสั่ง UPDATE เป็นดังนี้

UPDATE FROM<ชื่อไฟล์> ON<คีย์> ADD<รายชื่อของฟิลด์> REPLACE<รายชื่อของฟิลด์>

หมายเหตุ ก่อนใช้คำสั่งนี้ไฟล์ทั้งสองไฟล์จะต้องทำการเรียงมาก่อน (pre-sorted) ตามคีย์ฟิลด์

สร้างไฟล์ใหม่คือ WARRUP ตามคำสั่งต่อไปนี้

MACHINTYPE C, 2

PURPRICE N, 10,2

CHARGE N , 10,2

แล้วใส่เรคอร์ดต่อไปนี้ลงไป

DU 30000.00 100.00

KB 35000.00 200.00

จากนั้นทำตามคำสั่งต่อไปนี้

USE WARRANTY

INDEX ON MACHINTYPE TO TYPEIN

USE WARRANTY INDEX TYPEIN

UPDATE FROM WARRUP ON MACHINTYPE REPLACE PURPRICE ADD CHARGE

```
. USE WARRANTY
. INDEX ON MACHINTYPE TO TYPEIN
. USE WARRANTY INDEX TYPEIN
. UPDATE FROM WARRUP ON MACHINTYPE REPLACE PURPRICE ADD CHARGE
:
:
. LIST
00010  951283 998111 DU VIMONSILP LTD.      30000.00 11/05/83   100.00
00011  954873 998717 DU BOON THAIYONT       90000.99 12/01/83     0.00
00012  971981 997514 DU SOOTSUWAN           95000.00 11/10/83     0.00
00013  976121 998114 DU SAHAPANISH          89090.99 11/11/83     0.00
00006  750793 767231 KB P.NAKORN LTD.       35000.00 11/10/83   810.01
00007  759110 764115 KB SAHAPANISH          31000.50 11/12/83   620.01
00008  798717 765211 KB SIAM PANISH         31000.95 11/29/83   620.01
00009  875179 768811 KB HOME ELECTRONIC     32000.91 10/05/83     0.00
00014  978110 998998 KB SOOTSUWAN           30000.00 11/30/82   600.00
00001  546798 567891 PT BOON THAIYONT       19005.95 12/10/83   380.11
00002  546799 576888 PT BOON THAIYONT       19006.25 12/11/83   380.12
00003  557314 569811 PT HOME ELECTRONIC      7510.98 11/29/83   150.21
00004  557333 569411 PT VIMONSILP LTD.      17500.50 11/12/83   390.01
00005  558111 568791 PT VIMONSILP LTD.       7610.98 11/29/83   152.21
00015  985731 998432 PT PATANAPAN            5000.99 11/12/83     0.00
```

ลองใช้คำสั่ง LIST เพื่อจะดูว่าทุก ๆ เรคอร์ดจะแสดงผลออกมาเป็นอย่างไร จะเห็นว่าข้อมูลในไฟล์ทั้งสองจะถูกทำให้ย้อนไปเรคอร์ดแรก ณ ที่จุดเริ่มต้นใหม่ จากนั้นจะมีการเปรียบเทียบฟิลด์ซึ่งถ้าเหมือนกันโดยตลอดแล้ว ข้อมูลจากส่วนของไฟล์ WARRUP จะถูกใช้ได้ 2 ทางคือ จะรวมกันแบบตัวเลขแล้วเก็บไว้ในฐานข้อมูลที่ใช้ขณะนั้นหรืออาจจะถูกนำไปแทนที่ในไฟล์ที่ใช้อยู่แต่ถ้าฟิลด์ MACHINTYPE นั้นไม่แมทช์กัน เรคอร์ดทั้ง 2 นั้นจะได้รับการสกิ๊ปไป

ดังนั้นจะเห็นว่าคำสั่ง UPDATE ข้างบนนี้จะรวมข้อมูลทั้งหมดของราคาใหม่ ใน DU และ KB ซึ่งระบุไว้ในไฟล์ที่ทำการ UPDATE และจะรวมค่า 100 เข้าไปในไฟล์เริ่มแรก

```
.
. CREATE
ENTER FILENAME: WARRUP
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD   NAME,TYPE,WIDTH,DECIMAL PLACES
```

```
001       MACHINTYPE,C,2
002       PURPRICE,N,10,2
003       CHARGE,N,10,2
004
INPUT DATA NOW? Y


.
. LIST OFF
DU     30000.00        100.00
KB     35000.00        200.00
.
. LIST
00001  DU    30000.00        100.00
00002  KB    35000.00        200.00
.
. DISPLAY STRUCTURE
STRUCTURE FOR FILE:  WARRUP.DBF
NUMBER OF RECORDS:   00002
DATE OF LAST UPDATE: 12/12/83
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001     MACHINTYPE    C     002
002     PURPRICE      N     010     002
003     CHARGE        N     010     002
** TOTAL **               00023
.
. DISPLAY STRUCTURE
STRUCTURE FOR FILE:  WARRUP.DBF
NUMBER OF RECORDS:   00002
DATE OF LAST UPDATE: 12/12/83
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001     MACHINTYPE    C     002
002     PURPRICE      N     010     002
003     CHARGE        N     010     002
** TOTAL **               00023
```

การทำรายงานจากฐานข้อมูล

คำสั่งสำหรับเตรียมรายงานคือ REPORT ไม่ว่าจะเป็นบนจอภาพหรือจะพิมพ์ลงกระดาษผลที่ออกมาจะเป็นข้อมูลที่เก็บอยู่ในไฟล์ที่ใช้ขณะนั้น คำสั่ง REPORT จะต้องระบุคอลัมน์นำ (title column) โดยบอกความกว้างของคอลัมน์พร้อมทั้งหัวกระดาษที่จะต้องพิมพ์ในแต่ละฟิลด์ โดยกำหนดขึ้นในขณะทำรายงานนี้ การสร้างรายงานด้วยวิธีนี้เป็นวิธีที่สร้างตามรูปแบบที่กำหนด

การสร้างรายงานอย่างง่าย

1. ต้องเลือกดูว่าจะทำรายงานของระบบข้อมูลในไฟล์ใด สมมติว่าเป็น WARRANTY พิมพ์คำว่า USE WARRANTY

REPORT FORM WARRSUM TO PRINT

(จากจุดนี้ไปดีเบสทูจะแสดงผลออกมาในลักษณะตัวพิมพ์หนา และค่าที่ต้องการจะใส่ทั้งหมดต้องอยู่ในวงเล็บ) และให้ทำตามที่จะแสดงต่อไปนี้

```
.
. USE WARRANTY
. REPORT FORM WARRSUM TO PRINT
ENTER OPTIONS,M=LEFT MARGIN,L=LINES/PAGE,W=PAGE WIDTH
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: WARRANTY SUMMARY
DOUBLE SPACE REPORT? (Y/N) N
ARE TOTALS REQUIRED? (Y/N) Y
SUBTOTALS IN REPORT? (Y/N) N
COL      WIDTH,CONTENTS
001      12,CARDNUM
ENTER HEADING: CARDNUMBER;**********
002      11,SERIALNUM
ENTER HEADING: SERIAL NO;*********
003      9,MACHINTYPE
ENTER HEADING: PRODUCT;*******
```

```
004      25,USRCOMPY
ENTER HEADING: USER COMPANY;************
005      10,PURPRICE
ENTER HEADING: PRICE;*****
ARE TOTALS REQUIRED? (Y/N) Y
006      8,DATE
ENTER HEADING: DATE;****
007      10,CHARGE
ENTER HEADING: CHARGES;*******
ARE TOTALS REQUIRED? (Y/N) Y
008
```

เมื่อใส่รายละเอียดต่าง ๆ ของรายงานแล้วกดคีย์ RET ในขณะ เมื่อดีเบสทูหยุดที่หมายเลขฟิลด์อันต่อไปและเรายังไม่ได้คีย์ข้อความใดเข้าไป แสดงว่าเป็นการเริ่มให้ดีเบสทำรายงาน จากนั้นดีเบสทูจะเริ่มต้นทำงานตามรายละเอียดที่กำหนดให้และจะทำตามข้อมูลที่สั่ง ถ้าใส่ข้อมูลให้ไปไม่ถูกต้องเมื่อต้องการหยุดการทำรายงานนี้ให้กด ESC

รูปของคำสั่ง REPORT เป็นดังนี้

```
REPORT FORM { <ชื่อของรูปแบบ> } <scope> FOR { <ข้อความ>    } TO PRINT
            { <formname>     }             { <expression> }
```

ความหมายของ TO PRINT หมายถึงว่าจะส่งผลทั้งหมดแสดงออกทาง PRINTER (ในขณะเดียวกันจะปรากฏบนจอภาพด้วย) และ scope จะหมายถึงทั้งหมดไฟล์ถ้าไม่ระบุให้แน่ชัดลงไป

การสร้างรายงานนี้ สามารถสร้างการรวมตัวเลขในคอลัมน์ใดก็ได้ หรือจะแยกทำการรวมเป็นกลุ่มก่อน (subtotal) แล้วค่อยรวมยอดสุดท้ายอีกก็ได้ ลักษณะของรายงานจึงเป็นการนำเอาข้อมูลในแต่ละฟิลด์ที่มีอยู่ในไฟล์ออกมาพิมพ์ ถ้าเป็นตัวเลขก็ยังรวมยอดทบไว้ได้ การจัดวางรูปหัวกระดาษหรือจำนวนหน้าก็สามารถเซทได้ จากตัวอย่างที่ป้อนให้นี้สามารถที่จะสร้างรายงานได้ ดังแสดงได้ต่อไปนี้

WARRANTY SUMMARY

CARDNUMBER	SERIAL NO	PRODUCT	USER COMPANY	PRICE	DATE	CHARGES
**********	*********	*******	************	*****	****	*******
546798	567891	FT	BOON THAIYONT	19005.95	12/10/83	380.11
546799	576888	PT	BOON THAIYONT	19006.25	12/11/83	380.12
557314	569811	PT	HOME ELECTRONIC	7510.98	11/29/83	150.21
557333	569411	PT	VIMONSILP LTD.	19500.50	11/12/83	390.01
558111	568791	PT	VIMONSILP LTD.	7610.98	11/29/83	152.21
750793	767231	KB	P.NAKORN LTD.	30500.77	11/10/83	610.01
759110	764115	KB	SAHAPANISH	31000.50	11/12/83	620.01
798717	765211	KB	SIAM PANISH	31000.95	11/29/83	620.01
875179	768811	KB	HOME ELECTRONIC	32000.91	10/05/83	0.00
951283	998111	DU	VIMONSILP LTD.	91000.88	11/05/83	0.00
954873	998717	DU	BOON THAIYONT	90000.99	12/01/83	0.00
971981	997514	DU	SOOTSUWAN	95000.00	11/10/83	0.00
976121	998114	DU	SAHAPANISH	89090.99	11/11/83	0.00
978110	998998	KB	SOOTSUWAN	30000.00	11/30/82	600.00
** TOTAL **				592230.65		3902.69

การสร้างรายงานโดยใช้การรวมข้อมูลกลุ่มย่อย (SUBTOTAL)

หากต้องการจะสร้างรายงานจากไฟล์ชื่อ WARRANTY เพื่อที่จะหาค่า total charge ในแต่ละส่วนของฟิลด์ USRCOMPY เราจำต้องใช้คำสั่ง REPORT ในลักษณะที่ต่างจากที่กล่าวมาแล้วดังนี้

```
. USE WARRANTY INDEX USRIN
. REPORT FORM WARREP TO PRINT
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH W=65
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: CHARGE ON EACH USER COMPANY
DOUBLE SPACE REPORT? (Y/N) N
ARE TOTALS REQUIRED? (Y/N) Y
SUBTOTALS IN REPORT? (Y/N) Y
ENTER SUBTOTALS FIELD: USRCOMPY
SUMMARY REPORT ONLY? (Y/N) N
EJECT PAGE AFTER SUBTOTALS? (Y/N) N
ENTER SUBTOTAL HEADING: TOTAL WARRANTY CHARGE ON
COL      WIDTH,CONTENTS
001      10,DATE
ENTER HEADING: DATE;====
002      9,MACHINTYPE
ENTER HEADING: PRODUCT;=======
003      10,PURPRICE
ENTER HEADING: PRICE;=====
ARE TOTALS REQUIRED? (Y/N) Y
004
```

การใช้ REPORT นี้จึงเป็นการสร้างรูปฟอร์มที่ต้องการพิมพ์และเก็บรูปฟอร์มนี้ไว้ในไฟล์ได้ การใช้งาน REPORT เป็นวิธีการที่ไม่ยุ่งยากและสามารถจัดการให้พิมพ์หมายเลขหน้าและหัวกระดาษต่าง ๆ ได้ แต่อย่างไรก็ดีการกำหนดรูปแบบหัวกระดาษนอกจากใช้ REPORT ยังมีวิธีการอื่นอีกซึ่งสามารถเขียนสั่งงานได้ในรูปแบบโปรแกรมซึ่งจะทำการยกตัวอย่างให้ดูต่อไป และสำหรับตัวอย่างที่เขียนขึ้นมานี้ เมื่อลองแสดงผลจะได้ดังนี้

```
PAGE NO. 00001

             CHARGE ON EACH USER COMPANY

   DATE       PRODUCT      PRICE
   ====       =======      =====

* TOTAL WARRANTY CHARGE ON BOON THAIYONT
12/10/83    PT           19005.95
12/11/83    PT           19006.25
12/01/83    DU           90000.99
** SUBTOTAL **
                        128013.19
```

```
* TOTAL WARRANTY CHARGE ON HOME ELECTRONIC
11/29/83    PT              7510.98
10/05/83    KB             32000.91
** SUBTOTAL **
                           39511.89

* TOTAL WARRANTY CHARGE ON P.NAKORN LTD.
11/10/83    KB             30500.77
** SUBTOTAL **
                           30500.77

* TOTAL WARRANTY CHARGE ON SAHAPANISH
11/12/83    KB             31000.50
11/11/83    DU             89090.99
** SUBTOTAL **
                          120091.49

* TOTAL WARRANTY CHARGE ON SIAM PANISH
11/29/83    KB             31000.95
** SUBTOTAL **
                           31000.95

* TOTAL WARRANTY CHARGE ON SOOTSUWAN
11/10/83    DU             95000.00
11/30/82    KB             30000.00
** SUBTOTAL **
                          125000.00

* TOTAL WARRANTY CHARGE ON VIMONSILP LTD.
11/12/83    PT             19500.50
11/29/83    PT              7610.98
11/05/83    DU             91000.88
** SUBTOTAL **
                          118112.36

** TOTAL **
                          592230.65
```

การใช้คำสั่ง COUNT SUM และ TOTAL

เราสามารถหารายละเอียดหรืออาจจะหาผลรวมของข้อมูลได้โดยอาศัย COUNT, SUM และ TOTAL เช่น สมมติว่าเราต้องการจะหาค่าเฉลี่ยของเงินที่ต้องจ่ายไปในการประกัน (ในส่วนของ WARRANTY) นั่นก็คือเมื่อ MACHINTYPE = "DU" ในกรณีนี้เราไม่จำเป็นต้องใช้รายงานผลแบบธรรมดา แต่เราต้องการทราบว่ามีจำนวน MACHINTYPE กี่ชนิดที่เป็น DU และต้องการหาค่าผลรวมของรายจ่ายทั้งหมดจะเห็นว่าค่าเฉลี่ยก็คือ รายจ่ายรวม/จำนวน DU ทั้งหมด ซึ่งทำได้โดยใช้คำสั่ง SUM และ COUNT ดังนี้

```
.
. USE
. USE WARRANTY
. COUNT ALL FOR MACHINTYPE = "PT" TO NUM
COUNT = 00005
. SUM CHARGE ALL FOR MACHINTYPE = "PT" TO TCHARGE
 1452.66
. STORE TCHARGE/NUM TO AVERG
 290.53
. DISPLAY MEMORY
NUM          (N)     5
AVERG        (N)   290.53
TCHARGE      (N)   1452.66
** TOTAL **      03 VARIABLES USED  00018 BYTES USED
. LIST
00001   546798 567891 PT BOON THAIYONT         19005.95 12/10/83   380.11
00002   546799 576888 PT BOON THAIYONT         19006.25 12/11/83   380.12
00003   557314 569811 PT HOME ELECTRONIC        7510.98 11/29/83   150.21
00004   557333 569411 PT VIMONSILP LTD.        19500.50 11/12/83   390.01
00005   558111 568791 PT VIMONSILP LTD.         7610.98 11/29/83   152.21
00006   750793 767231 KB P.NAKORN LTD.         30500.77 11/10/83   610.01
00007   759110 764115 KB SAHAPANISH            31000.50 11/12/83   620.01
00008   798717 765211 KB SIAM PANISH           31000.95 11/29/83   620.01
00009   875179 768811 KB HOME ELECTRONIC       32000.91 10/05/83     0.00
00010   951283 998111 DU VIMONSILP LTD.        91000.88 11/05/83     0.00
00011   954873 998717 DU BOON THAIYONT         90000.99 12/01/83     0.00
00012   971981 997514 DU SOOTSUWAN             95000.00 11/10/83     0.00
00013   976121 998114 DU SAHAPANISH            89090.99 11/11/83     0.00
00014   978110 998998 KB SOOTSUWAN             30000.00 11/30/82   600.00
```

:

คำสั่ง COUNT

รูปทั่วไปของคำสั่ง COUNT คือ

COUNT <SCOPE> FOR {<ข้อความ> / <EXPRESSION>} TO <ตัวแปร>

คำสั่งนี้จะทำการนับจำนวนในเรคอร์ดในไฟล์ที่ใช้นั้นถ้าใช้ส่วนของ FOR ด้วยการนับจะนับเฉพาะตามจำนวนของเรคอร์ดที่สอดคล้องกับข้อความ (EXPRESSION) เท่านั้น และถ้ารวมส่วนของ TO เข้าด้วย ตัวเลขจำนวนเต็มที่นับได้จะนำไปเก็บในหน่วยความจำที่ตั้งขึ้นมาเป็นตัวแปร (MEMORY VARIABLE)

คำสั่ง SUM

รูปทั่วไปของคำสั่ง SUM คือ

SUM {<FIELD> / <SCOPE> FOR <ข้อความ>} TO <MEMVAR LIST>

คำสั่งนี้จะทำการรวมตัวเลขที่แสดงหรืออ้างถึงในข้อความ (EXPRESSION) ในไฟล์นั้น ทั้งนี้จะขึ้นกับส่วนของ SCOPE และส่วนของ FOR ด้วย

คำสั่ง TOTAL

มีรูปแบบดังนี้ (จะเห็นได้ว่าคล้ายกับ SUBTOTAL)

TOTAL ON <KEY> TO <DATABASE FIELDS LIST> FOR <EXPRESSION>

คำสั่ง TOTAL และ SUBTOTAL มีส่วนที่เหมือนกัน TOTAL ก็เป็นฟังก์ชันในคำสั่ง REPORT เช่นเดียวกัน แต่ที่ต่างกันคือ TOTAL นี้จะมีการนำผลที่ได้ไปเก็บไว้ในฐานข้อมูลแทนที่จะพิมพ์ออกมา

หมายเหตุ ฐานข้อมูลที่ใช้กับคำสั่งนี้จะต้องเรียงลำดับคำมาก่อนหรือต้อง INDEX มาก่อนลองใช้คำสั่งต่อไปนี้

```
.
.
. USE WARRANTY
. LIST STRU
STRUCTURE FOR FILE:   WARRANTY.DBF
NUMBER OF RECORDS:    00014
DATE OF LAST UPDATE:  00/00/00
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001     CARDNUM       C     006
002     SERIALNUM     C     006
003     MACHINTYPE    C     002
004     USRCOMPY      C     015
005     PURPRICE      N     010    002
006     DATE          C     008
007     CHARGE        N     008    002
** TOTAL **               00056
. INDEX ON USRCOMPY TO USRIN
00014 RECORDS INDEXED
.
.
```

```
.
. USE WARRANTY INDEX USRIN
. TOTAL ON USRCOMPY TO SUMMARY FIELD PURPRICE,CHARGE
00007 RECORDS COPIED
.
.
. USE SUMMARY
. LIST ALL
00001   546798 567891 PT BOON THAIYONT      128013.19 12/10/83   760.23
00002   557314 569811 PT HOME ELECTRONIC     39511.89 11/29/83   150.21
00003   750793 767231 KB P.NAKORN LTD.       30500.77 11/10/83   610.01
00004   759110 764115 KB SAHAPANISH         120091.49 11/12/83   620.01
00005   798717 765211 KB SIAM PANISH         31000.95 11/29/83   620.01
00006   971981 997514 DU SOOTSUWAN          125000.00 11/10/83   600.00
00007   557333 569411 PT VIMONSILP LTD.     118112.36 11/12/83   542.22
. LIST STRUCTURE
STRUCTURE FOR FILE:  SUMMARY.DBF
NUMBER OF RECORDS:   00007
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD        NAME        TYPE WIDTH   DEC
001       CARDNUM       C    006
002       SERIALNUM     C    006
003       MACHINTYPE    C    002
004       USRCOMPY      C    015
005       PURPRICE      N    010    002
006       DATE          C    008
007       CHARGE        N    008    002
** TOTAL **                00056
. USE
.
.
.
```

ความหมายของการกระทำในกรณีนี้คือ ดีเบสทูจะเปิดไฟล์อินเด็กซ์ USRIN ซึ่งก็จะเรียงลำดับ USRCOMPY ไว้ก่อนแล้ว ในคำสั่งที่ 2 คือ TOTAL นี้มีความหมายถึงการนำเอาข้อมูล PURPRICE ของ USRCOMPY เดียวกันมาบวกกันและนำ CHARGE ที่มีชื่อ USRCOMPY เหมือนกันบวกกันแล้วเก็บไว้ในไฟล์ชื่อ SUMMARY ดังนั้นการเก็บในไฟล์ SUMMARY จึงเก็บ USRCOMPY, PURPRICE, CHARGE โดย PURPRICE และ CHARGE จะเป็นค่าที่เป็นผลมาจากการรวมกันมาแล้ว การ USE SUMMARY และ LIST จะได้ข้อมูลที่เกิดการรวมด้วย TOTAL มาแล้ว

สังเกตว่าการใช้ TOTAL นี้ในฐานข้อมูลจะต้องเรียงลำดับในคีย์ที่ต้องการหามาแล้ว (ในที่นี้จะเรียง USRCOMPY) หากไม่จัดเรียงก็ต้องใช้ในรูปอินเด็กซ์ไฟล์

10

การใช้คอมมานด์ไฟล์

คอมมานด์ไฟล์ คือ ไฟล์ที่รวบรวมเอาชุดคำสั่งของดีเบสทูมาเรียงกันไว้ เพื่อที่ว่าเราสามารถที่จะใส่ข้อความคำสั่งได้ซ้ำแล้วซ้ำอีก โดยการระบุชื่อของคอมมานด์ไฟล์ก็เป็นการเรียกใช้ได้เลยแทนที่จะพิมพ์ชุดของคำสั่งใหม่อีกครั้ง อันที่จริงในการสร้างคอมมานด์ไฟล์ขึ้นมาเสมือนว่าเรากำลังทำการเขียนโปรแกรมคอมพิวเตอร์อยู่นั่นเอง

ในการสร้างคอมมานด์ไฟล์ขึ้นมานั้น เราจะใส่ชุดของคำสั่งเรียงกันลงไป และทำการเก็บคำสั่งทั้งหมดนี้อยู่ในไฟล์ที่บ่งบอกลักษณะด้วย .CMD เราอาจจะสร้างไฟล์โดยอาศัยโปรแกรมเวิร์ดโปรเซสซิ่งหรืออิดิเตอร์ ดีเบสทูจะให้เราใช้วิธีของอิดิเตอร์หากเราใส่คำสั่งต่อไปนี้

MODIFY COMMAND <ชื่อไฟล์>

การสร้างคอมมานด์ไฟล์อย่างง่าย

สมมติว่าเราต้องการจะสร้างคอมมานด์ไฟล์อย่างง่าย เพื่อใช้ในการนับจำนวน MACHIN-TYPE ที่เราระบุรายละเอียด และหาค่าเฉลี่ยของรายจ่ายการประกันในแต่ละชนิดของเครื่องจักร ในที่นี้เราจะเริ่มด้วยการใช้คำสั่งต่อไปนี้

```
. MODIFY COMMAND
ENTER FILENAME: AVERAGE
```

จากนั้นดีเบสทูจะทำการล้างจอภาพ ซึ่งขณะนี้เราสามารถเขียนอะไรลงไปในลักษณะของการแก้ไขทางจอภาพ ดังที่กล่าวมาแล้ว โดยให้คำสั่งต่อไปนี้อยู่ในไฟล์ชื่อ AVERAGE เมื่อพิมพ์โปรแกรมหรือ

```
* THIS COMMAND FILE IS TO CALCULATE THE AVERAGE CHARGE FOR WARRANTY
* ON A PRODUCT SPECIFIED BY THE USER
USE WARRANTY
STORE "   " TO PRD
@ 5,15 SAY "WE WANT TO FIND THE AVERAGE WARRANTY CHARGE"
@ 6,15 SAY "FOR A PARTICULAR MACHINE TYPE FOR YOU"
@ 9,15 SAY "PLEASE ENTER A MACHINE TYPE : " GET PRD
READ
```

```
@ 9,15 SAY "PLEASE ENTER A MACHINE TYPE :PT:"
IF PRD = "DU" .OR. PRD = "PT" .OR. PRD = "KB"
     COUNT ALL FOR MACHINTYPE = PRD TO NUM
     SUM CHARGE ALL FOR MACHINTYPE = PRD TO TCHARGE
     STORE TCHARGE/NUM TO AVER
     ERASE
     @ 10,15 SAY "AVERAGE WARRANTY CHARGES ON " + PRD
     @ 11,15 SAY "IS"+STR(AVER,10)
ELSE
     @ 10,15 SAY " ** ERROR:PRODUCT NOT FOUND ON FILE ** "
ENDIF
```

โครงสร้างโปรแกรมข้างต้นเสร็จแล้วใช้ CTRL W เพื่อเข้าสู่ดีเบสทู จากนั้นก็พร้อมที่จะทำการรันโปรแกรม AVERAGE ได้แล้วโดยใช้คำสั่ง

DO AVERAGE

เมื่อใช้คำสั่ง DO นี้ ดีเบสทูจะหาจาก CMD ไฟล์ตามชื่อที่กำหนดมา ทำงาน การทำงานจะกระทำเรียงทีละคำสั่ง หากพบคำสั่งใดพิมพ์ผิดก็จะปรากฏข้อความว่า SYNTAX ERROR เช่น ชื่อฟิลด์ MACHINTYPE เราพิมพ์เป็น MACHINE TYPE ดีเบสทูตรวจสอบแล้วก็จะปรากฏข้อความให้แก้ดังนี้

```
. DO AVERAGE

*** SYNTAX ERROR ***
                          ?
     COUNT ALL FOR MACHINE TYPE = PRD TO NUM
CORRECT AND RETRY (Y/N)? Y
CHANGE FROM :E T
CHANGE TO   :T
     COUNT ALL FOR MACHINTYPE = PRD TO NUM
MORE CORRECTIONS (Y/N)? N
```

ในการเรียกโปรแกรมให้ทำงานนี้ ถ้าอยู่ในตัวดีเบสทูจะใช้ DO แต่ถ้าหากว่าเราเริ่มเปิดเครื่อง และถ้าหากยังอยู่ในระบบซีพีเอ็มที่มีเครื่องหมาย A>อยู่ให้ใช้คำสั่ง "dbase AVERAGE" แทนได้เลย ในกรณีนี้ดีเบสทูจะทำการเรียกโปรแกรมตามชื่อที่กำหนดมาทำงานทันทีเลย หากโปรแกรมที่ป้อนเข้าไปดังได้กล่าวแล้วไม่มีข้อผิดพลาด สิ่งที่ปรากฏบนจอจะเป็นดังนี้

```
                    WE WANT TO FIND THE AVERAGE WARRANTY
                    CHARGE FOR A PARTICULAR MACHINE TYPE
                    FOR YOU
                    PLEASE ENTER A MACHINE TYPE :PT:
COUNT = 00005
 1452.66
 290.53


                    AVERAGE WARRANTY CHARGES ON PT
                    IS          290
```

คำสั่ง @ ......SAY........

คำสั่งนี้จะช่วยให้เราสามารถเลือกดูได้จากจอภาพว่า เราต้องการจะนำข้อมูลจากส่วนใดมาแสดงที่ตำแหน่งที่เราต้องการ รูปแบบของคำสั่งจะเป็นดังนี้

@ <COORDINATE> SAY <"PROMPT"> GET <ตัวแปร>

ด้วยคำสั่งนี้จะวางค่าของตัวอักษรในจุด coordinate ที่ต้องการ (ที่มุมบนซ้ายของ CRT จะถือเป็นจุด 0,0 หรือ "home")

SAY สามารถเลือกได้ตามใจชอบ และนอกจากจะใช้ SAY สำหรับแสดงข่าวสารข้อความ (message) แล้ว ยังสามารถใช้แสดงนิพจน์ได้คือ

@ COORDINAGE SAY EXPRESSION

คำสั่ง GET และ SAY

คำสั่งที่ใช้นี้จะต้องสามารถขยายการแสดงผลเพื่อแสดงค่าของตัวแปรที่กำหนด ค่าของตัวแปรที่จะอ้างถึงกับคำสั่ง GET และ SAY นั้น จะต้องได้รับการกำหนดให้มีตัวแปรชื่อนี้ก่อนหรือหาค่าได้ก่อนคำสั่ง GET นั้น นั่นคือเหตุผลที่ทำไมเราถึงต้องเก็บค่า BLANK ในตัวแปร PRD ก่อนที่จะใช้คำสั่ง GET

คำสั่ง READ

คำสั่งนี้จะทำให้เราสามารถใส่ข้อมูลให้กับตัวแปรได้ โดยการอ่านข้าทางคีย์บอร์ดหรือจากที่ใดที่หนึ่ง การใช้ READ นี้จะคู่กับคำสั่ง GET กล่าวคือการอ่านข้อมูลจะอ่านจากตำแหน่งที่ GET นั่นเอง

สำหรับเครื่องหมาย " * " จะบอกเพียงว่าข้อความตามหลังทั้งหมดนั้นเป็นเพียงคำอธิบายซึ่งไม่มีผลต่อการทำงานของโปรแกรม ลักษณะของ * ก็เหมือนกับคำสั่ง REM ในภาษาเบสิคนั่นเอง

การเลือกและการตัดสินใจในดีเบสทูกับคำสั่ง IF...ELSE...ENDIF

การตัดสินใจอย่างง่ายโดยหากมีแต่การตัดสินใจเพียงอย่างเดียวตามเงื่อนไขที่กำหนด เราจะไม่ใช้คำสั่ง ELSE แต่จะใช้ดังนี้

```
IF CONDITION
     DO COMMAMD(S)
ENDIF
```

การตัดสินใจโดยมีทางเลือก 2 ทาง ในกรณีนี้จะใช้รูปแบบคำสั่งดังนี้

```
IF CONDITION(S)
     DO COMMAND(S)
ELSE
     DO COMMAND(S)
ENDIF
```

เครื่องคอมพิวเตอร์จะทำการคำนวณคำสั่งทั้งหมด จนในที่สุดจะข้ามมาทำคำสั่งที่ตามหลัง ENDIF

CONTROL SEQUENCE รูปอื่น

```
DO WHILE CONDITION(S)
    DO COMMAND(S)
ENDDO
```

ในขณะที่เงื่อนไขที่เราระบุเป็นจริง (ทางลอจิก) คำสั่งทั้งหมดที่อยู่ในช่วงของคำสั่ง DO จะได้รับการประมวลผลการทำงานของคำสั่งนี้จะมีลักษณะคล้ายกับลูป โดยถ้าเงื่อนไขยังตรงก็จะลูปต่อไป

ลองใส่คำสั่งต่อไปนี้

```
. MODIFY COMMAND RETRIEVE

USE WARRANTY INDEX USRIN
STORE "                 " TO NAME
@ 4,15 SAY "ENTER THE KEY " GET NAME
READ
FIND &NAME
DO WHILE NAME $ USRCOMPY
         DISPLAY
         SKIP
ENDDO
```

เมื่อป้อนโปรแกรมเสร็จเรียบร้อยแล้วลองมาทำความเข้าใจกับโปรแกรมโดยลองพิจารณาดูทีละบรรทัด หลังจากเข้าใจได้แล้วลองเอ็กซีคิวโปรแกรมนี้ด้วยการคีย์ข้อความ

```
. DO RETRIEVE
```

สิ่งที่ปรากฏบนจอคือข้อความที่บอกให้ป้อนข้อมูลที่เป็นคีย์ ในที่นี้คือ BOON THAIYONT สังเกตดูว่าการกำหนดช่องสำหรับคีย์นี้ไม่จำเป็นต้องกำหนดกว้างไป อาจจะนำเฉพาะตัวหน้า ๆ มาตรวจสอบก็ได้

```
ENTER THE KEY :BOON THAIYON:
```

เมื่อป้อนคีย์แล้วสิ่งที่ได้คือเครื่องจะค้นหาข้อมูลตามเงื่อนไขดังกล่าวแล้วพิมพ์ผลลัพธ์ออกมา

```
00001  546798 567891 PT BOON THAIYONT      19005.95 12/10/83  380.11
RECORD: 00002
00002  546799 576888 PT BOON THAIYONT      19006.25 12/11/83  380.12
RECORD: 00011
00011  954873 998717 DU BOON THAIYONT      90000.99 12/01/83    0.00
RECORD: 00003
```

โดยสรุปจะเห็นได้ว่า ลักษณะของโปรแกรมนี้ใช้ไฟล์ชื่อ USRIN ซึ่งเป็นไฟล์ชนิดอินเด็กซ์และให้เราสร้างตัวแปรชื่อ NAME โดยกำหนดให้เป็นที่ว่างก่อน จากนั้นเครื่องพิมพ์ ENTER THE KEY และให้ราคีย์ข้อความที่ต้องการเรียกค้นหาในไฟล์ เมื่อ READ แล้วข้อความที่คีย์ก็จะเก็บในตัวแปรชื่อ NAME ค่า

สั่ง FIND & NAME หมายถึงการ FIND ค่าตามตัวแปร NAME เช่น ถ้าคีย์ BOON THAIYONT คำสั่งนี้ก็เสมือนกับว่า FIND BOON THAIYONT

การใช้ไฟล์ฐานข้อมูล 2 ไฟล์

ในขั้นต้นนี้จะแนะนำให้รู้จักกับคำสั่งพื้นฐานในการใช้ดีเบสทู ซึ่งคำสั่งต่าง ๆ เหล่านี้ใช้เป็นหลักในการสร้างฐานข้อมูล แก้ไขเรคอร์ด เพิ่มเติมเรคอร์ด การทำรายงาน การแสดงข้อความในเรคอร์ด การสร้างคีย์ และการเรียงลำดับของข้อมูล ดังเช่นตัวอย่างที่จะกล่าวถึงนี้เป็นการสร้างฐานข้อมูลที่ชื่อไฟล์ DEALERS ซึ่งมีโครงสร้างดังต่อไปนี้

```
STRUCTURE FOR FILE:  DEALERS.DBF
NUMBER OF RECORDS:   00015
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD     .  NAME        TYPE WIDTH   DEC
001       DEALER        C     015         หมายถึง ชื่อตัวแทนขายประกัน
002       ADDRESS       C     030         หมายถึง ที่อยู่ตัวแทนขายประกัน
003       CHANGWAT      C     010
004       AREA          C     005         หมายถึง รหัสพื้นที่
005       CONTACT       C     003         หมายถึง เงื่อนไข
006       STAFFNUM      N     003         หมายถึง จำนวนพนักงาน
** TOTAL **                 00067
```

เราไม่จำเป็นจะต้องใส่ข้อมูลในขณะนี้ ดังนั้นเราจะพิมพ์ N เมื่อมีคำถามที่ปรากฏขึ้นว่าจะใส่ข้อมูลหรือไม่ ต่อมาเราจะสร้างไฟล์ฐานข้อมูลอีกหนึ่งไฟล์ ชื่อ *WARRANTY* ซึ่งมีโครงสร้างดังต่อไปนี้

```
STRUCTURE FOR FILE:  WARRANTY.DBF
NUMBER OF RECORDS:   00015
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD        NAME        TYPE WIDTH   DEC
001       CARDNUM       C     005        หมายเลขการ์ด
```

002	DSERIALNUM	C	005	รหัสตัวแทน
003	PSERIALNUM	C	005	รหัสเครื่องจักร
004	KSERIALNUM	C	005	รหัส K
005	MACHINTYPE	C	007	ชนิดของเครื่องจักร
006	USRCOMPY	C	015	ชื่อบริษัทที่ทำประกัน
007	DEALER	C	015	ชื่อตัวแทนขายประกัน
008	DATEPUR	C	008	วันที่ขายประกัน
** TOTAL **			00066	

ไฟล์ที่เก็บข้อมูลของฐานข้อมูลทั้งสองนี้จะได้รับการสร้างขึ้น เราสามารถที่จะใส่ข้อมูลลงในไฟล์ฐานข้อมูลทั้งสองนี้ได้โดยใช้คำสั่ง

ในการป้อนข้อมูลเข้าในไฟล์นั้น อาจทำได้โดยใช้คำสั่ง USE DEALERS แล้วตามด้วย APPEND ซึ่งก็จะปรากฏรูปแบบของฟอร์มการป้อนข้อมูลตามที่ได้กล่าวมาแล้ว ในที่นี้จะทำการสร้างไฟล์สมมติซึ่งเป็นชื่อตัวแทนขายประกันตามเขตต่าง ๆ โดยแสดงที่อยู่ จังหวัดและเขต รวมทั้งเงื่อนไขการติดต่อและจำนวนพนักงานลองทำการใส่ข้อมูลต่อไปนี้

00001	THAI PAIBOON	1/4 BANMAI MUENG DIST.	CHONBURI	A0001	L01	10
00002	SIAM PAIBOON	93/5 CHAREANMUENG	BANGKOK	A0001	L00	28
00003	SKULTHAI CO,LTD	155 SOI SENASARID	NONTABURI	A0001	L00	6
00004	PAISAL SILP CO.	199 VISETKUL	TRANG	A0004	L00	11
00005	CITY^S SPARCK	16 UBONSAK MUENG DIST.	UBON	A0003	L01	8
00006	D.T.C. SUPPLY	11 RONGARUN WANGTONG	PITSANULOK	A0002	L01	13
00007	COMMERCIAL CO.	13/9 SUANYAI SAMPAYA	SONGKLA	A0004	L01	9
00008	MITTHI INSURE	90/5 RAMBUTTREE CHAK	BANGKOK	A0001	L00	22
00009	CHUSENG HUAT L.	83/3 POONIMIT BUMRUNG	CHAENGMAI	A0002	L00	11
00010	GENERAL EQUIP.	525 PORNSAWANG SUKHUMVIT	RAYONG	A0003	L00	12
00011	HOSE OF SIAM	28 MUKSALA DEEWAN	LAMPANG	A0002	L00	10
00012	DELCO INSURE	346 TROK CHAN YANNAWA	BANGKOK	A0001	L00	15
00013	INTER-MARKETING	364 PRAHOLYOTIN PAYAO	CHENGRAI	A0002	L01	20
00014	N.S.D.L.P.	115 SIRICHINDA CHAI	BANGKOK	A0001	L01	20
00015	RANGSIT	15/8 SONGKRANVITEE	LOPBUREE	A0001	L01	12

ในทำนองเดียวกันสำหรับไฟล์ WARRANTY เราก็จะทำการป้อนข้อมูลชุดใหม่ ซึ่งประกอบด้วยเลขรหัสต่าง ๆ และลูกค้าที่ขายประกันได้ (USRCOMPY) โดยจะมีตัวแทนขายประกัน และวันที่ขายประกันประกอบอยู่ด้วย ให้ทำการป้อนข้อมูลต่อไปนี้เพื่อจะนำข้อมูลในไฟล์ WARRANTY และ DEALERS มาใช้งานร่วมกัน

00001	25163	41825	36415	43152	KU	VIMOLSIL	THAI PAIBOON	02/12/83
00002	25056	43815	36241	46166	DU	SRISAWANG	RANGSIT	05/09/83
00003	25041	46715	38921	49116	KU	SRISAWANG	THAI PAIBOON	08/11/83
00004	24115	48731	39511	40318	DU	SAHASILP	GENERAL EQUIP.	02/05/83
00005	23116	49371	3670	40215	MU	NANAPAN	MITTHAI INSURE	10/07/83
00006	25416	49111	35201	41026	KT	BONSILPCHAI	HOUSE OF SIAM	23/08/83
00007	26438	43252	33511	41152	DU	MAHACHAK	DELCO INSURE	25/05/83
00008	24532	42138	32025	43251	DU	TAE KUAN	RANGSIT	13/05/83
00009	26531	41312	33028	41123	KU	THAI PLASTIC	THAI PAIBOON	11/11/83
00010	25316	44567	32053	48721	MT	SIAMPANICH	MITTHAI INSURE	12/09/83
00011	25438	49008	35026	46525	HK	SIRIYONT	RANGSIT	03/06/83
00012	43251	48320	37092	43211	DU	NGAMCHAI	RANGSIT	03/09/83
00013	24528	40531	32031	41112	MI	BANYAI	N.S.D.L.P.	08/08/83

การเพิ่มเติมคำสั่ง

คำสั่ง DISPLAY

ในบางครั้งเราต้องการแสดงผลว่ามีไฟล์อะไรบ้าง โดยที่ไม่ต้องการเข้าสู่ตัวซีพีเอ็ม ในการใช้ DIR เราก็สามารถทำได้โดยใช้ คำสั่ง DISPLAY

```
. DISPLAY FILE ON A

DATABASE FILES   # RCDS    LAST UPDATE
DEALERS   DBF    00015     00/00/00
WARRANTY  DBF    00015     00/00/00
CHECKREG  DBF    00004     00/00/00
CUST      DBF    00010     15/15/15
Q         DBF    00000     00/00/00
CHECKREC  DBF    00000     00/00/00
```

ซึ่งจะแสดงชื่อไฟล์ทั้งหมดที่สร้างขึ้นโดยดีเบสทู ในดิสค์ไดร์ฟ A และถ้าใช้คำสั่ง DISPLAY FILES ON A LIKE * .CMD

```
.
.
. DISPLAY FILE ON A LIKE *.CMD

MENU    .CMD    NEWENTR .CMD    BALANCE .CMD    DPROGRES.CMD
DTEST   .CMD    CANCELS .CMD    RE      .CMD    REPORT  .CMD
A       .CMD    E       .CMD    W       .CMD    Z       .CMD
AA      .CMD
```

กรณีนี้จะแสดงชื่อไฟล์ในดิสค์ไดร์ฟ A ซึ่งเป็นไฟล์ชนิด .CMD ไฟล์ชนิดต่าง ๆ เช่น .TXT. .FRM .MEM หรืออื่น ๆ ก็จะสามารถแสดงขึ้นมาดูได้เช่นกัน ซึ่งการแสดงผลไฟล์ต่าง ๆ นี้จะมีลักษณะคล้ายคำสั่ง DIR ในซีพีเอ็ม

คำสั่ง BROWSE

เป็นคำสั่งที่มีประสิทธิภาพสูงมากในการแก้ไขข้อมูล ข้อมูลจำนวน 19 เรคอร์ดจะแสดงผลขึ้นบนจอ จำนวนฟิลด์จะแสดงขึ้นให้มากที่สุดเท่าที่จะมากได้ในแต่ละบรรทัด จอภาพจะเปรียบเสมือนหน้าต่างที่มองเข้าไปในไฟล์ฐานข้อมูล ซึ่งเราสามารถที่จะเลื่อนตำแหน่งเคอร์เซอร์ไปข้างหน้าหรือถอยหลังได้ ดังนั้นข้อมูลต่าง ๆ ก็จะสามารถแก้ไขได้โดยสะดวก

คีย์ที่ใช้ควบคุมการทำงานต่าง ๆ ที่จะใช้ในคำสั่ง BROWSE มีดังต่อไปนี้

CTRL b — เลื่อนจอหน้าต่างไปทางซ้าย 1 ฟิลด์
CTRL z — เลื่อนจอหน้าต่างไปทางขวา 1 ฟิลด์
CTRL c, a — กลับไปยังฟิลด์เดิมที่ผ่านมา
CTRL x, f — เลื่อนไปยังฟิลด์ถัดไป

CTRL q – ออกจากการทำงานของคำสั่ง BROWSE โดยที่ไม่เปลี่ยนแปลงข้อมูลเดิม
CTRL w – ออกจากการทำงานของคำสั่ง BROWSE โดยเปลี่ยนแปลงข้อมูลตามที่แก้ไข

หมายเหตุ การใช้คำสั่งดังกล่าวข้างบนนี้ใช้ได้โดยการกดคีย์ CONTROL (CTRL) และตัวอักษรพร้อม ๆ กัน เช่น กด "CTRL" พร้อมกับอักษร "b" หมายถึง การเลื่อนจอหน้าต่างไปทางซ้าย 1 ฟิลด์ เป็นต้น

ฟังก์ชันที่ใช้ในฐานข้อมูล

ในการใช้งานในโปรแกรม ผู้ใช้สามารถใช้ฟังก์ชันต่าง ๆ ที่ได้รับการกำหนดขึ้นโดยดีเบสทู ได้หลายฟังก์ชัน โดยผู้ใช้สามารถนำมาใช้เป็นเงื่อนไขทางลอจิก หรือประกอบในประโยคคำสั่งได้โดยตรง รูปแบบฟังก์ชันที่ใช้กันมากได้แก่

? (นิพจน์ หรือ ตัวแปร) – เป็นคำสั่งที่กำหนดให้เครื่องคอมพิวเตอร์ทำการพิมพ์ข้อมูลที่เป็นค่าของนิพจน์ หรือ ค่าตัวแปรนั้น ออกมา

EOF – เป็นตัวฟังก์ชันบอกการสิ้นสุดหรือจบไฟล์แล้ว หรือยัง ค่าที่ได้ของฟังก์ชันนี้จะเป็นค่า (logic) ว่า "จริง" (TRUE) หรือ "เท็จ" (FALSE)

$ (ค่าตัวแปรสตริง, ตำแหน่งเริ่มต้น, ความยาว) – เป็นฟังก์ชันสตริง โดยกำหนดจุดเริ่มต้นของสตริงและความยาวของตัวสตริงที่จะนำมา เช่น STORE "ABCDEFCH" TO CHAR, ? $(CHAR, 3,4) ผลลัพธ์ที่ได้คือ CDEF หรือหมายถึงการนำเอาสตริง (ข้อความ) จากค่าตัวแปร CHAR เริ่มจากตัวที่ 3 นับไป 4 ตัว

! (สตริง) – เป็นฟังก์ชันบอกว่าสตริงเป็นอักษรตัวใหญ่เท่านั้น ถึงแม้ว่าค่าเดิมจะเป็นอักษรตัวเล็ก โปรแกรมจะเปลี่ยนเป็นตัวใหญ่ เช่น

```
STORE "ADcdEFgh" TO CHAR
?  !(CHAR)
```

ผลลัพธ์ที่ได้คือ ABCDEFGH

LEN(สตริง) – เป็นฟังก์ชันที่จะบอกถึงความยาวของสตริงว่ามีกี่ตัวอักษร เช่น ? LEN(CHAR) ผลลัพธ์ที่ได้คือ 8

รายงานที่สร้างขึ้นจากไฟล์ที่แตกต่างกัน

ในหัวข้อที่ผ่านมา เราได้สร้างฐานข้อมูลขึ้นมา 2 ไฟล์ ซึ่งมีชื่อไฟล์ว่า WARRANTY และ DEALERS ซึ่งทำให้เราสามารถหลีกเลี่ยงการเขียนข้อมูลที่ซ้ำ ๆ กันได้ โดยจัดข้อมูลให้อยู่ในไฟล์ 2 ไฟล์ที่แตกต่างกัน เช่น ADDRESS ของ DEALERS จะไม่ต้องเขียนซ้ำในไฟล์ WARRANTY

สมมติว่าเราต้องการจะผลิต DEALERS' PROGRESS REPORT เราจำเป็นจะต้องมีรายละเอียดของตัวแทนผู้มาติดต่อทุก ๆ คน รวมทั้งลูกค้าด้วย หรือกล่าวอีกนัยหนึ่งคือ เราต้องการจะทำรายงานโดยใช้ฟิลด์ DEALER, ADDRESS จากไฟล์ DEALER และใช้ฟิลด์ USRCOMPY, MACHINTYPE และ DATE-

PUR จากไฟล์ WARRANTY แต่ในไฟล์ WARRANTY นั้นจะมีตัวแทนขาย (Dealers) ซึ่งสามารถขายประกันเครื่องจักรหลายชนิดให้กับลูกค้าหลายคนได้ จึงมีข้อมูลได้หลายเรคอร์ด แต่ในไฟล์ DEALER นั้น จะมีชื่อตัวแทนเพียงตัวเดียวสำหรับลูกค้าแต่ละคน ดังนั้นในการทำรายงานนี้ เราจะเอาชื่อของผู้แทนติดต่อขาย (Dealers'name) จากฟิลด์ DEALER ในไฟล์ DEALERS' ไปเข้าคู่กับทุก ๆ ชื่อลูกค้าใน DEALER ฟิลด์ ของลูกค้าในไฟล์ WARRANTY

ดังตัวอย่างเช่นในไฟล์ WARRANTY มีชื่อตัวแทน THAI PAIBOON อยู่ 3 เรคอร์ดเราต้องการสร้างรายงานว่า ตัวแทนแต่ละรายขายประกันได้กี่รายมีชื่อลูกค้าที่ขายได้เป็นใครบ้าง ในกรณีของตัวแทน THAI PAIBOON ก็ขายให้ลูกค้าชื่อ VIMOLSIL, SRISAWANG และ THAI PLASTIC รายงานจึงประกอบด้วย ชื่อตัวแทน ที่อยู่และลูกค้าที่ขายได้

ทบทวนเกี่ยวกับการทำรายงาน

ก่อนที่จะเข้าใจการจัดรายงานในสองไฟล์ ขอให้เข้าใจการเรียกไฟล์ในโหมดปฐมภูมิและทุติยภูมิ ซึ่งการเรียกจะใช้ SELECT เป็นตัวเลือกและถ้าเลือกโหมดไหนก็จะคงโหมดนั้นก่อนลองมาทำความเข้าใจกับคำสั่ง REPORT โดยทบทวนจากของเดิมโดยใช้คำสั่งดังนี้

```
. USE DEALERS
. SELECT SECONDARY
. USE WARRANTY
. REPORT
ENTER REPORT FORM NAME: DREPT
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH W=65
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: DEALER'S PROGRESS REPORT
DOUBLE SPACE REPORT? (Y/N) N
ARE TOTALS REQUIRED? (Y/N) Y
SUBTOTALS IN REPORT? (Y/N) Y
ENTER SUBTOTALS FIELD: P.DEALER
SUMMARY REPORT ONLY? (Y/N) N
EJECT PAGE AFTER SUBTOTALS? (Y/N) N
ENTER SUBTOTAL HEADING: **DEALER**
COL      WIDTH,CONTENTS
001      30,S.USRCOMPY
ENTER HEADING: CUSTOMER NAME
002      20,S.MACHINTYPE
ENTER HEADING: MACHINE TYPE
003      10,S.DATEPUR
ENTER HEADING: DATE
004
```

ในกรณีที่เราเริ่มเลือก DEALERS เป็นไฟล์ปฐมภูมิ และเลือก WARRANTY เป็นไฟล์ทุติยภูมิ แล้วสร้าง REPORT ไฟล์ จากไฟล์ทุติยภูมิ โดยมี SUBTOTAL ใช้ฟิลด์จากไฟล์ปฐมภูมิชื่อ DEALER (สังเกตการใช้ P. DEALER) ดังนั้นเมื่อเสร็จสิ้น ดีเบสจะสร้างรายงานทั้งหมดจากไฟล์ทุติยภูมิ *ซึ่งยังไม่ถูกต้อง* ผลที่ได้ดังตัวอย่างข้างบนนี้

```
PAGE NO. 00001
                    DEALER'S PROGRESS REPORT

       CUSTOMER NAME                    MACHINE TYPE          DATE

* **DEALER** THAI PAIBOON
VIMOLSIL                                KU                    02/12/83
SRISAWANG                               DU                    05/09/83
SRISAWANG                               KU                    08/11/83
SAHASILP                                DU                    02/05/83
NANAPAN                                 MU                    10/07/83
BONSILPCHAI                             KT                    23/08/83
MAHACHAK                                DU                    25/05/83
TAE KUAN                                DU                    13/05/83
THAI PLASTIC                            KU                    11/11/83
SIAMPANICH                              MT                    12/09/83
SIRIYONT                                HK                    03/06/83
NGAMCHAI                                DU                    03/09/83
BANYAI                                  MI                    08/08/83
SAHAVIRIYA LTD.                         DU                    12/08/83
ROONGRUENGSIN                           KU                    09/09/83
```

สังเกตว่าชื่อตัวแทน THAI PAIBOON ขายประกันได้เฉพาะ VIMOLSIL, SRISAWANG และ THAI PLASTIC เท่านั้น แต่รายงานนี้พิมพ์ออกมาทุกเรคอร์ดซึ่งเป็นเรื่องที่เราไม่ต้องการ แต่อย่างไรก็ตามการสร้างรายงานนี้ดีเบสจะสร้างไฟล์ชื่อ DREPT ไว้เป็นไฟล์ชนิด .FRM ซึ่งสามารถเรียกมาใช้ตามเงื่อนไขต่าง ๆ ได้ภายหลัง และเราจะนำเอาข้อดีในส่วนนี้มาเขียนในโปรแกรมเพื่อสั่งงานให้ดีเบสสร้างรายงานตามที่เราต้องการได้

ในที่นี้จะสร้างโปรแกรม ชื่อ DPROGRESS รายละเอียดของการใช้ MODIFY COMMAND ได้กล่าวแล้วในบทที่ 10

ในกรณีนี้ใช้

MODIFY COMMAND DPROGRES

ดีเบสจะแสดงจอว่างแล้วเราคีย์โปรแกรมดังต่อไปนี้

```
SET TALK OFF
USE DEALERS
GO TOP
    DO WHILE .NOT. EOF
         STORE $(P.DEALER,1,15) TO DNAME
         SELECT SECONDARY
         USE WARRANTY
         SET HEADING TO 1983 DEALERS PROGRESS REPORT
         REPORT FORM DREPT FOR DNAME = $(S.DEALER,1,15)
         SELECT PRIMARY
         SKIP
    ENDDO
```

เสร็จแล้วใช้คำสั่ง CTRL-W เพื่อเก็บโปรแกรมไว้ในจานแม่เหล็ก

จากนั้นเรียก DO DPROGRES ผลจะได้ดังต่อไปนี้

```
PAGE NO. 00001         1983 DEALERS PROGRESS REPORT

                  DEALER'S PROGRESS REPORT

        CUSTOMER NAME                  MACHINE TYPE        DATE

* **DEALER** THAI PAIBOON
VIMOLSIL                           KU                  02/12/83
SRISAWANG                          KU                  08/11/83
THAI PLASTIC                       KU                  11/11/83

PAGE NO. 00001         1983 DEALERS PROGRESS REPORT

                  DEALER'S PROGRESS REPORT

        CUSTOMER NAME                  MACHINE TYPE        DATE

* **DEALER** GENERAL EQUIP.
SAHASILP                           DU                  02/05/83

PAGE NO. 00001         1983 DEALERS PROGRESS REPORT

                  DEALER'S PROGRESS REPORT

        CUSTOMER NAME                  MACHINE TYPE        DATE

* **DEALER** DELCO INSURE
MAHACHAK                           DU                  25/05/83

PAGE NO. 00001         1983 DEALERS PROGRESS REPORT

                  DEALER'S PROGRESS REPORT

        CUSTOMER NAME                  MACHINE TYPE        DATE

* **DEALER** N.S.D.L.P.
BANYAI                             MI                  08/08/83

PAGE NO. 00001         1983 DEALERS PROGRESS REPORT

                  DEALER'S PROGRESS REPORT

        CUSTOMER NAME                  MACHINE TYPE        DATE

* **DEALER** RANGSIT
SRISAWANG                          DU                  05/09/83
TAE KUAN                           DU                  13/05/83
SIRIYONT                           HK                  03/06/83
NGAMCHAI                           DU                  03/09/83
```

คำสั่ง SELECT PRIMARY/ SELECT SECONDARY

คำสั่งนี้จะทำให้ดีเบสทูทำการเลือกข้อมูลที่เป็นไปได้มา 1 ข้อมูลจาก 2 ข้อมูลเพื่อมาทำการคำนวณในแต่ละคำสั่งผู้ใช้งานสามารถจะกระทำกับข้อมูล 2 ชุด ในวลาเดียวกันได้ เช่นเดียวกับคำสั่ง REPORT ในคอมมานด์ไฟล์

เมื่อดีเบสเริ่มทำงาน ส่วนที่เป็น PRIMARY จะเริ่มพร้อมที่จะทำงาน และรักษาสภาพนี้อยู่จนกว่าจะมีการ SELECT SECONDARY จากนั้นส่วน SECONDARY จะแอคตีฟอยู่จนกว่าส่วน PRIMARY จะใช้งานอีก (สั่งอีกครั้ง) เมื่อข้อมูลทั้งสองไฟล์ในฐานข้อมูลที่ใช้งาน ฟิลด์ที่เป็นตัวแปรจะสามารถแยกออกจากส่วนอื่นหรืออาจกล่าวว่านิพจน์ใด ๆ จะสามารถใช้ตัวแปรภายในขอบเขตของฐานข้อมูลได้ แต่สำหรับกรณีที่ชื่อฟิลด์ในส่วนที่สองเหมือนกัน ตัวแปรที่ใช้จึงจำต้องมีตัวนำหน้า (ในที่นี้คือ " P." และ " S.") เพื่อที่จะบอกว่าตัวแปรมาจากฐานข้อมูลใด

ในตัวอย่างของเรา DEALERS ถือเป็นฐานข้อมูลปฐมภูมิ และ WARRANTY เป็นข้อมูลทุติยภูมิ ทดสอบโดยการใช้ค่าทางลอจิก จะพบว่าการทำงานของระบบฐานข้อมูลทุติยภูมิและปฐมภูมิเป็นอย่างไร

คำสั่ง SET TALK OFF

เมื่อใช้คำสั่งนี้แล้ว ผลที่แสดงบนจอภาพทั้งหมดจะถูกลบออก การแสดงผลจากตัวที่ไม่ต้องการได้ (Undesirable) เช่น หมายเลขเรคอร์ด สำหรับแต่ละคำสั่ง SKIP จะไม่ปรากฏบนจอภาพ คำสั่งสำหรับหยุด SET TALK คือ ON

คำสั่ง SET HEADING TO

คำสั่งนี้จะทำให้คำสั่ง SET เก็บค่าตัวแปรสตริงไว้ และจะพิมพ์บรรทัดบนออกมาเป็นชื่อหัวของรายงานซึ่งความยาวของสตริงอาจยาวได้ถึง 60 ตัวอักษร ดังนั้น "1982 DEALERS PROGRESS REPORT" จะถูกพิมพ์เป็นชื่อหัว

คำสั่ง SUBSTRING

หมายถึงฟังก์ชัน SUBSTRING ซึ่งมีรูปแบบดังได้กล่าวไว้แล้ว ในกรณีนี้คือค่าของสตริงตัวแปรชื่อ S.DEALER เริ่มจากตัวที่ 1 จำนวน 15 ตัว ในคอมมานด์ไฟล์เราใช้ฟังก์ชันสับสตริงเพื่อที่จะหา 15 ตัวอักษรแรก จากตัวอักษรของ P. DEALER เพื่อทำการเปรียบเทียบเพื่อจะหลีกเลี่ยงอุปสรรคต่าง ๆ เช่น ลืม COMMA หรือลืม full stop อันจะทำให้เกิดข้อความผิดพลาดได้

การใช้อินเด็กซ์ไฟล์ในฐานข้อมูลทุติยภูมิ

เราใช้คำสั่ง REPORT และ SUBTOTAL เพื่อความสะดวกในการหารายละเอียดของลูกค้าที่ติดต่อกับ DEALER ในไฟล์ WARRANTY ถ้าสมมติว่าไฟล์ WARRANTY นั้นใหญ่มากและค่าเรคอร์ดทั้งหมดเป็นค่าที่ใส่โดยสุ่มเข้าไป คำสั่ง REPORT จะทำงานช้ามาก ซึ่งเราอาจจะสามารถทำให้เร็วขึ้นได้โดยอาศัยคำสั่ง INDEX ดังที่กล่าวมาแล้ว มาทดลองว่า INDEX จะสามารถช่วยการทำงานเร็วขึ้นอย่างไร โดยใส่คำสั่งต่อไปนี้

```
MODIFY COMMAND DTEST
```

เรากำลังจะสร้างคอมมานด์ไฟล์ เพื่อจะให้ผลออกมาดังต่อไปนี้

```
  DO DTEST
00007  26438 43252 33511 41152 DU     MAHACHAK        DELCO INSURE       25/05/83
00004  24115 48731 39511 40318 DU     SAHASILP        GENERAL EQUIP.     02/05/83
00013  24528 40531 32031 41112 MI     BANYAI          N.S.D.L.P.         08/08/83
00002  25056 43815 36241 46166 DU     SRISAWANG       RANGSIT            05/09/83
00008  24532 42138 32025 43251 DU     TAE KUAN        RANGSIT            13/05/83
00011  25438 49008 35026 46525 HK     SIRIYONT        RANGSIT            03/06/83
00012  43251 48320 37092 43211 DU     NGAMCHAI        RANGSIT            03/09/83
00001  25163 41825 36415 43152 KU     VIMOLSIL        THAI PAIBOON       02/12/83
00003  25041 46715 38921 49116 KU     SRISAWANG       THAI PAIBOON       08/11/83
00009  26531 41312 33028 41123 KU     THAI PLASTIC    THAI PAIBOON       11/11/83
```

เมื่อจบแล้วใช้ CTRL-W ลองใช้คำสั่ง DO DTEST ดู ผลที่ได้จะเป็นคล้ายกับรายงานที่เราได้สร้างมาแล้ว แต่คราวนี้พิมพ์ออกมาติดกันเป็นหมวดหมู่ตามการอินเด็กซ์ไว้ดังต่อไปนี้

```
A>TYPE DTEST.CMD
SET TALK OFF
USE WARRANTY
INDEX ON DEALER TO WDINX
USE DEALERS
INDEX ON DEALER TO DDINX
USE DEALERS INDEX DDINX
GO TOP
* DEALER - LOOP
DO WHILE .NOT. EOF
      STORE $(P.DEALER,1,15) TO DNAME
      SELECT SECONDARY
      USE WARRANTY INDEX WDINX
      FIND &DNAME
      * WARRANTY - LOOP
      DO WHILE DNAME = $(S.DEALER,1,15).AND..NOT.EOF
              DISPLAY
              SKIP
      ENDDO
      * END WARRANTY - LOOP
      SELECT PRIMARY
      SKIP
ENDDO
*END DEALERS LOOP
```

มาลองทดสอบดูว่าโปรแกรมนี้ทำงานได้อย่างไร

ลูปนอกวงการควบคุมของ PRIMARY ไฟล์ เช่น (dealers file) เมื่อใช้กับคำสั่ง SKIP loop นี้จะทำการตรวจสอบข้อมูลใน INDEXED DEALERS ไฟล์ ทีละตัว ๆ

ลูปในของการใช้การควบคุมของ SECONDARY ซึ่งก็คือ ไฟล์ WARRANTY และจะข้ามคำสั่งก่อนที่จะวนลูปเพื่อที่จะหาค่าของ indexed WARRANTY ไฟล์ทีละตัว ๆ

โดยทั่วไปแล้ว คอมมานด์ไฟล์จะทำงานโดยการทำเรคอร์ดจากไฟล์ DEALERS (และใช้ชื่อ

ของ DEALER) จากนั้นจะเปิดไฟล์ WARRANTY เป็นไฟล์ที่ 2 และจะทำการตรวจสอบเรคอร์ดแรกดูว่า แมทช์กับ DEALER ในฟิลด์ไหนโดยที่ไฟล์ WARRANTY นั้น ได้ทำการเรียงลำดับตัวอักษรในฟิลด์ DEALER แล้ว ดังนั้นเราจึงสามารถให้เครื่องอ่านเรคอร์ดต่อไปในไฟล์ WARRANTY จนกว่าจะไม่สามารถแมทช์กับ DEALER ได้ หรือจนกว่าจะจบไฟล์นั้นหลังจากที่เราได้แสดงดูเรคอร์ดทั้งหมดใน WARRANTY ไฟล์ ซึ่งมีค่าเหมือนกับในฟิลด์ DEALER ในไฟล์ DEALERS จากนั้นเราจะไปทำการวนในลูปใหม่ และเลือกค่า PRIMARY ไฟล์ แล้วผ่านไปยังเรคอร์ดต่อไปในไฟล์ DEALERS (โดยใช้คำสั่ง SKIP) และหาไฟล์ WARRANTY อีกครั้งโดยใช้คำสั่ง FIND คำสั่งของคอมมานด์ไฟล์ เหล่านี้จะหยุดลงเพื่อ dealers ทั้งหมดในไฟล์ DEALERS จะผ่านทุกขั้นตอน

EOF (END OF FILE) ที่ใช้ทั้งใน PRIMARY และ SECONDARY ไฟล์ จะต้องตรวจสอบดูก่อนเพราะจะได้กันการคำนวณที่ผิดพลาดที่อาจเกิดขึ้นเมื่อคำนวณถึง EOF ในฐานข้อมูลทั้งสอง

หมายเหตุ ในคำสั่ง FIND หน่วยความจำที่เป็นตัวแปร (DNAME) จะถูกค้นหา เพราะฉะนั้น "&" จะต้องถูกแทนที่ก่อนตัวแปร DNAME

เราสามารถหาค่าผลรวมได้โดยใช้คำสั่ง FIND และ SKIP แต่เราสามารถทำให้เร็วขึ้นกว่านั้นได้ หากเราใช้ไฟล์ที่ index แล้วตามด้วยคำสั่ง DO WHILE LOOP

ฟังก์ชัน คำสั่งพิเศษ และการสั่งฟอร์แมทแสดงผล

ในบทนี้เราจะขอกล่าวถึงฟังก์ชันพิเศษอื่น ๆ เพิ่มเติม ที่จะช่วยทำให้โปรแกรมที่เราเขียนขึ้นโดยใช้ดีเบสทูนี้ มีประสิทธิภาพการใช้งานตรงตามจุดประสงค์ที่แตกต่างกันไปได้ดียิ่งขึ้น ชุดคำสั่งต่าง ๆ ที่จะสามารถนำไปใช้ในโปรแกรมคำสั่ง (Command file) ที่จะกล่าวถึงนี้ จะมีผลทำให้ผลลัพธ์ของการประมวลตามความต้องการของเรานั้นง่ายดายและตรงตามจุดประสงค์พิเศษยิ่งขึ้น

เพิ่มประสิทธิภาพของโปรแกรมด้วยฟังก์ชัน

ฟังก์ชันจะเป็นชุดคำสั่งที่จะทำงานตามจุดประสงค์เฉพาะอย่าง โดยจะให้ผลลัพธ์ที่ต้องการออกมาอาจจะคำนวณหรือประมวลผลมาจากนิพจน์ สมการ หรือตัวแปรที่กำหนด และส่วนใหญ่จะเป็นฟังก์ชันซึ่งผู้ใช้อาจจะเขียนโปรแกรมขึ้นเองได้โดยลำบาก ดังนั้นดีเบสทูจึงได้เตรียมชุดคำสั่งที่จะทำงานเป็นฟังก์ชันพิเศษเพื่อประมวลในเชิงคณิตศาสตร์ การเปรียบเทียบ หรือทำการประมวลผลทางด้านอักขระของตัวอักษร คำสั่งต่าง ๆ เหล่านี้เราอาจจะนำไปใช้โปรแกรมคำสั่ง (command file) หรืออาจจะทดลองดูโดยการใช้คำสั่ง ? โดยการพิมพ์เครื่องหมาย (?) และตามด้วยช่องว่าง (blank) หนึ่งช่อง แล้วจึงตามด้วยชุดคำสั่งต่าง ๆ เหล่านี้ ซึ่งจะได้ผลลัพธ์ออกมาตามลักษณะการทำงานของฟังก์ชันนั้น ๆ

ข้อสังเกต ข้อความในคำสั่งจะมีเครื่องหมายวงเล็บเปิด “(” และวงเล็บปิด “)” จะต้องพิมพ์ลงในคำสั่งในเวลาที่เราใช้งานด้วย

! (<ตัวแปร/ตัวอักษร>)

เป็นฟังก์ชันในการแปลงอักษรที่อยู่ภายในวงเล็บจากตัวอักษรตัวเล็กให้เป็นอักษรตัวใหญ่ นั่นคือ ฟังก์ชันนี้จะทำการเปลี่ยนตัวอักษร ‘a’..‘b’..‘z’ ให้กลายเป็น ‘A’..‘B’...‘Z’ ซึ่งเป็นตัวอักษรตัวใหญ่ตามลำดับส่วนตัวอักษรที่เป็นอักษรอื่น ๆ จะคงไว้อย่างเดิม การใช้ฟังก์ชันดังกล่าวนี้จะช่วยทำให้ง่ายต่อการค้นหาข้อมูล (database file) ซึ่งเก็บเป็นตัวอักษรตัวใหญ่เสมอ

TYPE (<นิพจน์>)

เป็นฟังก์ชันในการหาชนิดของข้อมูลที่คำนวณได้จากนิพจน์นั้น คำตอบที่ได้จากฟังก์ชันจะเป็นได้

เพียง 3 แบบ คือ เป็น C, N หรือ L ซึ่งจะบอกให้ทราบว่านิพจน์ดังกล่าวนี้เป็นชนิดตัวอักษร (Character) ตัวเลข (Numeric) หรือเป็นแบบลอจิก (logic) ตามลำดับ

INT (<ตัวแปร/นิพจน์>)

เป็นฟังก์ชันของเลขจำนวนเต็ม โดยทำการคำนวณที่อยู่ภายในวงเล็บซึ่งอาจจะเป็นตัวแปรหรือนิพจน์ที่ต้องมีการคำนวณเป็นเลขออกมาก่อนแล้วจะทำการตัดเลขทศนิยมนั้นทิ้ง คงเหลือแต่เลขจำนวนเต็มเป็นคำตอบของฟังก์ชันนี้

ยกตัวอย่างเช่น INT(23.86) จะได้ผลลัพธ์เป็น 23

หรือฟังก์ชัน INT(– 124.46X จะได้ผลลัพธ์เป็น – 123

ถ้าในกรณีที่เราต้องการผลลัพธ์เป็นเลขจำนวนเต็มที่มีการปัดเศษของทศนิยมตำแหน่งที่หนึ่ง เราอาจจะทำได้โดยการใช้ INT (ตัวเลข + 0.5) ซึ่งการทำงานของฟังก์ชันนี้จะมีการคำนวณผลบวกก่อนแล้วจึงตัดเลขหลังจุดทศนิยมทิ้ง ซึ่งก็จะได้ผลลัพธ์ของการปัดจุดทศนิยมที่เกิน 0.5 ขึ้นมาเป็นเลข 1 นั่นเอง

และถ้าในกรณีที่ต้องการปัดตัวเลขให้มีเหลือทศนิยมเพียง 1 ตำแหน่ง อาจจะใช้ฟังก์ชัน INT นี้ได้โดยการใช้ว่า INT (ตัวเลข *10 + 0.5)/10 ซึ่งฟังก์ชันดังกล่าวนี้จะปัดจุดทศนิยมในตำแหน่งที่ 2 ที่เกิน 0.05 ขึ้นมาอยู่เป็นตัวเลขที่มีทศนิยมเพียงตำแหน่งเดียว

VAL (<ตัวแปร/ตัวอักษร)

เป็นฟังก์ชันในการแปลงตัวอักษรให้เป็นตัวเลขจำนวนเต็ม ฟังก์ชันนี้จะทำการเปลี่ยนแปลงข้อมูลในเชิงตัวอักษร อาจจะประกอบด้วยเครื่องหมายบวก ลบ ตัวเลข และรวมทั้งเลขทศนิยมขนาด 1 ตำแหน่ง ให้กลายเป็นข้อมูลชนิดตัวเลขที่มีค่าเท่ากันและพร้อมที่จะนำไปคำนวณกันในเชิงคณิตศาสตร์ต่อไปได้ เช่น VAL('123') จะได้คำตอบเป็นข้อมูลตัวเลข 123 (เดิมตัวข้อมูล "123" จะเก็บในหน่วยความจำในลักษณะของรหัสตัวอักษรที่ไม่สามารถนำไปคำนวณในเชิงคณิตศาสตร์ได้) VAL(Job:Nmbr) จะให้คำตอบเป็นตัวเลข หาค่าของตัวแปร Job:Nmbr โดยทั่ว ๆ ไปเราอาจจะเก็บข้อมูลของหมายเลขของพนักงาน (Job Numbers) ในแบบข้อมูลอักขระ (string) แต่เมื่อเราต้องการนำข้อมูลนี้ไปคำนวณก็จำเป็นจะต้องแปลงให้เป็นรหัสแบบตัวเลขเพื่อการคำนวณต่อไปก่อนนั่นเอง

STR (<นิพจน์/ตัวแปร/ตัวเลข> , <ความยาว> , <จำนวนเลขหลังจุดทศนิยม>)

เป็นฟังก์ชันในการแปลงตัวเลขจำนวนเต็มให้กลายเป็นตัวอักษร (string) ซึ่งจะทำการแปลงข้อมูลที่อยู่ภายในตัวเล็บซึ่งอาจจะคำนวณได้จากนิพจน์หรือเป็นค่าตัวแปร หรือเป็นตัวเลขโดยตรงเลยให้กลายเป็นการเก็บข้อมูลแบบอักขระอักษร โดยที่เราจะสามารถกำหนดความยาวของอักขระที่ต้องการเปลี่ยนว่าเป็นจำนวนกี่ตัว และยังสามารถกำหนดจำนวนของตัวเลขหลังจุดทศนิยมที่ต้องการด้วย ค่าของตัวเลขที่บอกความยาวจะต้องกำหนดให้มากพอที่จะครอบคลุมจำนวนตัวเลขทั้งหมดรวมทั้งจุดทศนิยมด้วย ถ้าตัวเลขที่เราจะเปลี่ยนแปลงนั้นมีจำนวนตัวเลขน้อยกว่าที่เรากำหนดความยาว ผลลัพธ์ที่ได้ก็จะมีจำนวนช่องว่างเพิ่มขึ้นตามไปด้วย และในกรณีที่เราใช้ฟังก์ชันโดยไม่กำหนดตำแหน่งของตัวเลขหลังจุดทศนิยม เครื่องคอมพิวเตอร์จะถือว่าไม่มีเลขหลังจุดทศนิยม

LEN (<ตัวแปร/ตัวอักขระ>)

เป็นฟังก์ชันหาจำนวนของตัวอักษร โดยจะให้ผลลัพธ์ออกมาเป็นตัวเลขที่บอกจำนวนตัวอักษรทั้งหมดของตัวแปรหรือตัวอักขระที่กำหนดมานั้น ฟังก์ชันนี้จะใช้ประโยชน์ได้มากในการที่จะหาความยาวของตัวแปรเพื่อจะเตรียมเนื้อที่ในการเก็บข้อมูลต่อไป แต่ถ้าในกรณีที่เราใช้ฟังก์ชันนี้ในการหาความยาวของตัวแปรที่เป็นชื่อของฟิลด์ (field name) คำตอบที่ได้จะเป็นตัวเลขที่แสดงขนาดของความยาวของฟิลด์นั่น มิใช่จำนวนตัวอักษรภายในฟิลด์

$ (<นิพจน์/ตัวแปร/ตัวอักขระ> , <ตัวเลขแสดงตำแหน่งเริ่มต้น> , <ตัวเลขแสดงความยาว>)

เป็นฟังก์ชันที่เราเรียกว่า *substring function* ซึ่งจะหมายถึง ฟังก์ชันที่จะทำการดึงข้อความบางส่วนออกจากนิพจน์หรือตัวแปร หรือตัวอักษรภายในวงเล็บออกมาเพียงบางส่วนโดยการที่เราสามารถที่จะบอกตำแหน่งของอักษรตั้งแต่ตัวที่เท่าไรนับจากทางซ้ายมือ และยังกำหนดอีกว่าดึงอักษรนั้นมาเป็นจำนวนกี่ตัวด้วย

ยกตัวอย่างเช่น ถ้าเรามีตัวแปรที่ชื่อว่า <DATE> ได้ทำการเก็บข้อมูลในลักษณะของรหัสแบบอักขระเป็น "810823" แล้วเราใช้ฟังก์ชันว่า $ (DATE,5,2) คำตอบที่ได้จะมีค่าเท่ากับ "23" (ดึงข้อมูลจากตัวแปร DATE ตั้งแต่ตำแหน่งที่ 5 มาจำนวน 2 ตัวอักษร) ตัวแปรที่ได้เป็นคำตอบนี้ถ้าเราต้องการที่จะนำไปคำนวณต่อไป เราก็จำเป็นต้องแปลงให้เป็นรหัสของตัวเลขโดยใช้ฟังก์ชันดังนี้ว่า VAL($(DATE,5,2)) ซึ่งจะได้คำตอบเป็นตัวเลข 23

@ (<ตัวแปรตัวที่ 1/ ตัวอักษรชุดที่ 1> , <ตัวแปรตัวที่ 2/ตัวอักษรชุดที่ 2>)

เป็นฟังก์ชันในการค้นหาตัวอักษรที่เรียกว่า *Substring search function* จะเป็นการค้นหาข้อมูลว่าตำแหน่งที่เท่าไรที่ตัวอักขระชุดที่ 1 มีอยู่ในตัวอักขระในชุดที่ 2 ดังนั้นคำตอบที่ได้จะเป็นตัวเลขที่บอกตำแหน่งที่จะค้นหาดูว่ามีอักขระตัวที่ 1 อยู่ในตัวอักขระชุดที่ 2 ที่เหมือนกันเริ่มตั้งแต่ตำแหน่งที่เท่าไร ถ้าในกรณีที่ค้นหาแล้วไม่มีตรงกัน คำตอบที่ได้จะเป็น "0"
ยกตัวอย่างเช่น

@ ("WED", "MONTUEWEDTHUFRISATSUN")

คำตอบที่ได้จะเป็นตัวเลข 7

ข้อสังเกต ในกรณีที่เราต้องการจะทราบแต่เพียงว่ามีข้อมูลของตัวอักษรชุดที่ 1 อยู่ในตัวอักษรชุดที่ 2 หรือไม่เท่านั้น (โดยอาจจะไม่ต้องการรู้ว่าอยู่ตำแหน่งเท่าไร) เราอาจจะใช้ฟังก์ชันที่เรียกว่า *substring logical operator* คือตัวอักษรชุดที่ 1 $ ตัวอักษรชุดที่ 2 คำตอบที่ได้นี้จะเป็นแบบโลจิก กล่าวคือได้เป็น ใช่ *(Yes)* หรือ *(No)* เท่านั้น ''ACB'' $ ''DEFGH'' จะได้คำตอบเป็น NO (บางเครื่องจะให้เป็นตัวย่อว่า N หรือ F false)

CHR (<number>)

ตัวเลขที่อยู่ภายในวงเล็บจะเป็นตัวเลขแสดงรหัสแบบ "ASCII" เช่นตัวเลข 65 จะเป็นรหัส

แบบ ASCII นี้ของตัวอักษร A ดังนั้นถ้าเราใช้คำสั่งว่า ? CHR(65) แล้วเราก็จะได้ผลลัพธ์เป็นอักษร A นั่นเอง ในเครื่องไมโครคอมพิวเตอร์บางเครื่องนั้นรหัสของคำสั่งควบคุมการทำงานของเครื่องบางอย่างอาจจะแตกต่างกันไป เช่น ? CHR(26) จะเป็นการลบข้อความบนจอทิ้งทั้งหมด (clear screen) ในการควบคุมบางครั้งอาจจะรวมไปถึงการส่งรหัสไปควบคุมเครื่องพิมพ์ เพื่อกำหนดขนาดของตัวอักษร ขีดเส้นใต้ตัวอักษร การใช้ฟังก์ชันพิเศษเหล่านี้จำเป็นจะต้องศึกษาดูรายละเอียดของเครื่องนั้น ๆ ดูว่ามีการใช้งานอย่างไร เพราะเครื่องแต่ละยี่ห้ออาจจะแตกต่างกันไปบ้าง

& เป็นฟังก์ชันพิเศษที่เราเรียกว่า *macro substitution function* เมื่อใช้เครื่องหมายดังกล่าวนี้โดยการวางไว้ตำแหน่งหน้าสุดของชื่อตัวแปรหน่วยความจำ (memory variable name) dBASE II จะทำการนำค่าของตัวแปรนั้น ซึ่งควรจะเป็นตัวอักษรนั้นมาวางใจตำแหน่งนั้นแทนทันที ยกตัวอย่างเช่น

```
? 'Which file do you want to review ?"
ACCEPT TO Database
USE & Database
```

จากคำสั่งข้างบนนี้ในบรรทัดแรกคอมพิวเตอร์จะพิมพ์ข้อความในเครื่องหมายคำพูดออกมาแล้ว รอรับข้อมูลจากผู้ใช้ผ่านทางคีย์บอร์ด เพื่อรับข้อมูลเข้ามาเก็บในตัวแปรชื่อ Database ถ้าสมมติว่าเราป้อนว่า Test แล้วกดคีย์บอร์ด return คำสั่งต่อมาก็จะมีค่าเท่ากับ เราใช้คำสั่งว่า USE Test นั่นเอง

ถ้าเราใช้คำสั่งว่า STORE "Delete Record" TO D แล้วหลังจากนั้นเราจะสามารถที่จะลบ 'record" ที่ 5 ทิ้งได้โดยการใช้คำสั่งว่า &D 5 เป็นต้น

FILE (<"ชื่อไฟล์"/ตัวแปร/นิพจน์>)

เป็นฟังก์ชันให้ค้นหาดูว่าภายในดิสค์ขณะนี้มีไฟล์ที่มีชื่อเท่ากับที่กำหนดภายในวงเล็บหรือไม่ โดยคำตอบที่ได้จะเป็น *ไม่มี (false)* หรือ *มี (true)* ถ้าข้อความในวงเล็บเป็นชื่อไฟล์โดยตรง เราจะต้องเขียนไว้ภายในเครื่องหมายคำพูด แต่ถ้าเราใช้เป็นตัวแปรที่เก็บข้อมูลตัวอักษรก็ไม่จำเป็นต้องมีเครื่องหมายคำพูด ถ้าเราใช้คำสั่งว่า FILE "B:" + Database ก็จะเป็นการค้นหาชื่อไฟล์ที่ชื่อว่า Database ว่ามีอยู่ภายในดิสค์ช่อง B หรือไม่นั่นเอง

TRIM (ตัวแปรตัวอักษร)

เป็นฟังก์ชันที่จะทำการกำจัดช่องว่าง (blank) ทางขวามือที่อยู่ภายในตัวแปรที่ไม่ใช่ช่องว่างที่อยู่ระหว่างตัวอักษร

STORE TRIM (<ตัวแปร>) TO <ตัวแปรตัวใหม่>

เช่น ถ้าตัวแปรตัวหนึ่งมีชื่อว่า Name เก็บข้อมูล "Kaset" แล้วใช้คำสั่ง

STORE TRIM (Name) TO New

เราจะได้ตัวแปรที่ชื่อว่า New มีค่าของข้อมูลเท่ากัน "Kaset" (ไม่มีช่องว่างทางขวามือของตัวอักษร)

ชุดคำสั่งสำหรับการกำหนดคุณสมบัติพิเศษของดีเบสทู

ดีเบสทูได้ตระเตรียมชุดคำสั่งพิเศษเพื่อใช้ในการควบคุมการทำงานของระบบ เช่น มีการควบคุมระบบการทำงานในการติดต่อกับเครื่องพิมพ์ หรือการแสดงผลบนจอภาพ เป็นต้น คุณสมบัติในการทำงานของระบบตามคำสั่งต่าง ๆ นั้น เราจะกำหนดได้โดยการใช้ชุดคำสั่งของ SET ความจริงแล้วดีเบสทู ได้กำหนดคุณสมบัติของการทำงานในระบบควบคุมเหล่านี้ไว้ให้เหมาะสมอยู่แล้ว แต่อย่างไรก็ตามผู้ใช้ก็อาจจะควบคุมให้ได้คุณสมบัติตามวัตถุประสงค์ที่ต้องการได้โดยการเปลี่ยนแปลงคุณสมบัตินั้น ๆ ด้วยคำสั่ง SET ที่จะกล่าวถึงต่อไปนี้ได้

ชุดคำสั่ง SET ต่อไปนี้ขอให้สังเกตว่าคำสั่งที่ได้ขีดเส้นใต้ไว้นั้นจะเป็นคำสั่งที่ดีเบสทูได้ทำการกำหนดไว้เองโดยอัตโนมัติทุกครั้งที่เราเรียกโปรแกรมดีเบสทูขึ้นมาในระบบ ดังนั้นเราอาจจะต้องการเปลี่ยนแปลงได้ โดยการเลือกชุดคำสั่งที่จะเปลี่ยนแปลงคุณสมบัติของระบบตามความเหมาะสมดังนี้

SET TALK <u>ON</u>	– จะทำการแสดงผลเนื่องมาจากชุดคำสั่งอื่น ๆ ทุกคำสั่งจะไม่มีการ
OFF	แสดงผล
SET PRINT ON	– จะทำการส่งข้อความทุกอย่างที่แสดงผลที่จอภาพเนื่องจากคำสั่งเอาท์พุทไปแสดงออกที่เครื่องพิมพ์ด้วย
<u>OFF</u>	– ไม่มีการพิมพ์ที่เครื่องพิมพ์
SET CONSOLE <u>ON</u>	– แสดงผลออกมาบนจอภาพเมื่อมีคำสั่งที่เกี่ยวข้องกับการเอาท์พุท (output)
OFF	– ไม่มีการแสดงผลบนจอภาพ
SET SCREEN <u>ON</u>	– จะทำให้มีลักษณะการป้อนข้อมูลผ่านทางคีย์บอร์ดในแบบที่เรียกว่า *full screen operation* กล่าวคือ เราสามารถเลื่อนเครื่องหมายแสดงตำแหน่งการแสดงผลบนจอ (cursor) ไปที่ตำแหน่งใด ๆ บนจอภาพเพื่อการป้อนข้อมูลได้ ซึ่งจะเป็นประโยชน์มากเมื่อเราใช้คำสั่ง APPEND, EDIT, INSERT และคำสั่ง CREATE
OFF	– จะทำให้ไม่มีการป้อนข้อมูลแบบ full screen
SET FORMAT TO <u>SCREEN</u>	– เป็นคำสั่งที่จะทำการส่งผลลัพธ์ของคำสั่ง @ ไปแสดงผลบนจอภาพ
PRINT	– จะทำการส่งผลลัพธ์ของคำสั่ง @ ไปแสดงผลทางเครื่องพิมพ์
SET MARGIN TO 〈nnn〉	– เป็นคำสั่งที่จะกำหนดตำแหน่งคอลัมน์ทางซ้ายมือสุดที่เครื่องพิมพ์จะพิมพ์ โดยที่เราจะสามารถกำหนดให้ตัวเลข 〈nnn〉 มีค่าได้ไม่เกิน 254 หรือมีค่าได้ไม่เกินจำนวนตัวอักษรที่พิมพ์ได้บนเครื่องพิมพ์ที่ใช้งาน
SET RAW ON	– ผลลัพธ์เนื่องมาจากการใช้คำสั่ง DISPLAY หรือคำสั่ง LIST จะแสดงข้อมูลในแต่ละเรคอร์ด โดยไม่มีช่องว่างขึ้นอยู่ระหว่างเรคอร์ด เหล่านี้เลย
<u>OFF</u>	– คำสั่ง DISPLAY หรือคำสั่ง LIST จะให้ผลลัพธ์ที่มีช่องว่าง 1 ช่องขึ้นอยู่

		ในแต่ละเรคอร์ด ที่แสดงผลออกมาบนจอภาพนั้น
SET HEADING ON <string>	—	จะเป็นคำสั่งที่จะทำการเปลี่ยนข้อความหัวกระดาษ (heading) ที่เราอาจจะกำหนดมาก่อนหน้านี้ด้วยคำสั่ง REPORT ให้มีข้อความใหม่เป็นข้อความที่อยู่หลังคำสั่ง SET นี้
SET ECHO ON	—	ตัวคำสั่งทุก ๆ คำสั่งที่เราทำการเอ็กซิคิวท์จากไฟล์คำสั่งจะถูกนำมาแสดงผลบนจอภาพด้วย
<u>OFF</u>	—	ไม่มีการแสดงผลตัวคำสั่ง
SET EJECT ON	—	จะมีการเลื่อนหน้ากระดาษพิมพ์โดยอัตโนมัติเมื่อใช้คำสั่ง REPORT
<u>OFF</u>	—	ไม่มีการเลื่อนหน้ากระดาษแบบอัตโนมัติ
SET STEP ON	—	เมื่อหลังจากที่เราใช้คำสั่งนี้แล้วดีเบสทู จะทำคำสั่งแต่ละคำสั่งต่อมาแบบที่มีการหยุดรอหลังจากที่ทำงานเสร็จทีละหนึ่งคำสั่งแล้วทุกครั้ง ซึ่งคำสั่งนี้จึงมีประโยชน์ในการตรวจสอบหรือแก้ไขโปรแกรม
<u>OFF</u>	—	ทำงานตามคำสั่งเรื่อยไปอย่างปกติ
SET DEBUG ON	—	จะทำการส่งผลลัพธ์จากคำสั่ง ECHO และ STEP ไปแสดงผลออกทางเครื่องพิมพ์เท่านั้น
<u>OFF</u>	—	ส่งผลลัพธ์ของคำสั่ง ECHO และ STEP ไปแสดงที่จอภาพ
SET BELL <u>ON</u>	—	จะกำหนดให้มีเสียงคล้ายเสียงกระดิ่งดังขึ้นทุกครั้งที่เราป้อนข้อมูลจนเต็มฟิลด์ (field) แล้ว
OFF	—	ไม่มีเสียงดัง
SET COLON <u>ON</u>	—	จะกำหนดให้การแสดงผลของฟิลด์เมื่อเครื่องหมายจุดคู่ (colon:) แสดงมาบนจอเพื่อปิดหัวและท้าย เพื่อแสดงความกว้างของฟิลด์นั้น
OFF	—	ไม่มีการแสดงเครื่องหมายจุดคู่ (:)
SET CONFIRM ON	—	เมื่อเราป้อนข้อมูลไปจนเต็มขนาดของฟิลด์แล้ว จะยังมีการรอรับคีย์ RETURN จากผู้ใช้ด้วย เพื่อจะเป็นการบอกให้เลื่อนตำแหน่งการรับข้อมูลไปยังฟิลด์ถัดไป
<u>OFF</u>	—	จะไม่มีการรอรับคีย์บอร์ด RETURN เมื่อเราได้ป้อนข้อมูลเต็มฟิลด์แล้ว
SET CARRY ON	—	เมื่อเราใช้คำสั่ง APPEND เพื่อที่จะทำการป้อนข้อมูลเพิ่มในไฟล์ข้อมูลนั้น DBASE II จะแสดงผลข้อมูลในเรคอร์ดสุดท้ายของไฟล์เดิมก่อน แล้วจึงจะสามารถเลื่อนไปยังเรคอร์ดใหม่ที่จะทำการป้อนข้อมูลได้
<u>OFF</u>	—	เมื่อเราใช้คำสั่ง APPEND นั้น ดีเบสทูจะแสดงเรคอร์ดใหม่ (ซึ่งไม่มีข้อมูล) ที่ต่อท้ายไฟล์ข้อมูลเก่าทันที
SET INTENSITY <u>ON</u>	—	จะทำให้มีการแสดงผลในขณะที่อยู่ในโหมดการทำงานแบบ full-screen edit โดยแสดงตัวอักษรในแบบที่แสดงผลเป็นตัวหนังสือจาง ๆ

กว่าปกติ หรือเครื่องคอมพิวเตอร์บางเครื่องจะแสดงผลแบบ *inverse video* กล่าวคือจะได้ตัวหนังสือดำบนพื้นขาว จอภาพสีขาว แทนการแสดงผลตามปกติที่เป็นตัวหนังสือสีขาวบนพื้นสีดำนั่นเอง

OFF — จะไม่มีการแสดงผลดังกล่าว แต่จะแสดงผลแบบปกติธรรมดาบนจอภาพ

SET LINKAGE ON — เป็นคำสั่งที่จะทำให้การแสดงผลด้วยคำสั่ง LIST หรือ DISPLAY แล้วแสดงผลของข้อมูลในไฟล์ 2 ไฟล์ คือในไฟล์ปฐมภูมิ (Primary) และไฟล์ทุติยภูมิ (Secondary) ได้พร้อมกัน ทำให้ดูเหมือนว่าเรามีไฟล์ที่มีจำนวนฟิลด์ได้สูงสุด 64 ฟิลด์ และข้อมูลสูงสุดต่อ 1 เรคอร์ด เท่ากับ 2000 ไบท์

<u>OFF</u> — จะไม่มีการ LIST หรือ DISPLAY แบบดังกล่าวข้างบน โดยที่ไฟล์แบบธรรมดามีจำนวนฟิลด์สูงสุดได้ 32 ฟิลด์ และข้อมูลสูงสุดต่อ 1 เรคอร์ด มีเพียง 1000 ไบท์ (ต่ำกว่านี้ได้)

SET EXACT ON — เมื่อมีการเปรียบเทียบตัวแปรชนิดที่เก็บข้อมูลตัวอักษร 2 ตัวแปรว่าเท่ากันหรือไม่นั้น จะทำให้มีการเปรียบเทียบอย่างถูกต้องทุก ๆ ตัวอักษรในตัวแปรนั้น ยกตัวอย่างเช่น ถ้าต้องการเปรียบเทียบ "ABCD" = "AB" เราจะได้คำตอบเป็น "ไม่เท่ากัน" (False)

<u>OFF</u> — เมื่อมีการเปรียบเทียบตัวแปรตัวอักษรนั้น จะไม่สนใจขนาดความยาวของตัวอักษรว่าเท่ากันหรือไม่ก็ได้ เช่น ต้องการเปรียบเทียบ "ABCD" = "AB" เราจะได้คำตอบว่า "เท่ากัน" (True)

SET ESCAPE <u>ON</u> — จะอนุญาตให้มีการหยุดการทำงานของไฟล์คำสั่ง (command file) ได้เมื่อมีการกดคีย์ escape (ESC) โดยจะหยุดออกจากโปรแกรมมาเลยเมื่อกดคีย์ดังกล่าวนี้

OFF — จะไม่มีการเลิกล้มคำสั่งในไฟล์คำสั่ง ถ้าแม้จะกดคีย์ ESC

SET ALTERNATE TO 〈ชื่อไฟล์〉คำสั่งนี้จะทำการสร้างไฟล์บนดิสค์ให้มีชื่อไฟล์ตามที่เรากำหนดในคำสั่งนี้ โดยมีนามสกุลของไฟล์ (extension) ว่า .TXT (กล่าวคือ จะสร้างไฟล์ว่า "ชื่อไฟล์ที่กำหนด .TXT") เพื่อพร้อมที่จะรับข้อมูลหรือข้อความทุกอย่างที่แสดงอยู่บนจอภาพ โดยที่เราจะต้องเริ่มต้นการรับข้อความลงไฟล์นี้โดยใช้คำสั่งว่า *SET ALTERNATE ON* และเราอาจจะต้องการเปลี่ยนไฟล์ที่จะเก็บข้อมูลใหม่ได้โดยการใช้คำสั่งว่า SET ALTERNATE TO〈ชื่อไฟล์ใหม่〉 หลังจากที่เราบอกให้คอมพิวเตอร์บอกการเริ่มต้นการรับข้อมูลจากจอภาพด้วยคำสั่ง SET ALT ALTERNATE ON แล้ว เราจำเป็นต้องบอกการสิ้นสุดการรับข้อมูลลงไฟล์โดยจะใช้คำสั่งว่า *SET ALTERNATE OFF* ซึ่งความจริงจะสิ้นสุดการรับข้อมูลลงใน

ไฟล์นี้ได้เมื่อเราใช้คำสั่ง QUIT เพื่อออกจากโปรแกรม dBASE II ด้วยดังนั้นจะเห็นว่าคำสั่งเหล่านี้จะทำให้เราได้ไฟล์ที่เก็บข้อความที่เก็บอยู่ในไฟล์แบบที่เราอาจจะใช้โปรแกรมจำพวก *เวิร์ดโปรเซสเซอร์* *(Word Processor)* เพื่อวัดข้อความเพิ่มเติมในไฟล์เหล่านี้ได้เพื่อความเหมาะสมต่อไป

คำสั่งที่กำหนดการรับข้อมูลแบบ full-screen editing และกำหนดรูปฟอร์มการรับข้อมูลด้วยคำสั่ง @...SAY...GET...PICTURE

ระบบการจัดการฐานข้อมูล dBASE II มีคุณสมบัติที่มีประสิทธิภาพสูงมากในความสามารถสำหรับการรับข้อมูลในแบบ full-screen กล่าวคือ เลื่อนเครื่องหมาย cursor ไปที่ตำแหน่งที่ต้องการจะป้อนข้อมูลตำแหน่งก็ได้บนจอภาพ สำหรับการส่งผ่านข้อมูลไปแสดงที่จอภาพโดยใช้คำสั่งพิมพ์ (@) กล่าวคือ

@ 〈coordinates〉 SAY ['prompt'] GET 〈variable〉

จากคำสั่งดังกล่าวนี้เราจะสามารถที่จะส่งข้อความ (prompt) ซึ่งอยู่ภายในเครื่องหมายคำพูดหลังคำสั่ง SAY ไปยังตำแหน่งใดบนจอภาพได้ด้วยการกำหนดโดยใช้ตัวเลขแสดงบรรทัดที่ และตำแหน่งคอลัมน์ที่ (coordinate y, x) พร้อมกันนั้นอาจจะส่งค่าของตัวแปรออกไปแสดงผลด้วยได้ โดยที่ตัวแปรนั้นจะแสดงอยู่หลังคำสั่ง GET จะเห็นแล้วว่าคำสั่งดังกล่าวนี้จะเป็นชุดคำสั่งในการแสดงผลเอาท์พุทออกมา แต่อย่างไรก็ตามชุดคำสั่งนี้ซึ่งอาจจะนำมาต่อกันหลาย ๆ คำสั่งแล้วตามด้วยคำสั่ง READ จะกลับทำให้การทำงานของระบบกลายเป็นการที่พร้อมที่จะรับข้อมูลในลักษณะของ full screen editing ได้ทันที ถ้าเราลองเขียนไฟล์คำสั่งโดยใช้ชุดคำสั่งที่เป็นตัวอย่าง

```
.
.
STORE "........" TO MDATE
STORE "........" TO MBALANCE
STORE "........" TO MDRAW
STORE "........." TO MEVALUATE
@ 5,5 SAY "Set date MM/DD/YY" GET MDATE
@ 10,5 SAY "What is the balance?" GET MBALANCE
@ 15,5 SAY "How much is request" GET MDRAW
READ
ERASE
@ 5,5 SAY "Should we do an evaluation?" GET MEVALUATE
READ
```

จากคำสั่งดังกล่าวข้างบนนี้จะทำให้ผู้ใช้สามารถทำการป้อนข้อมูลลงในตัวแปร MDate, MBalance และ MDraw ในตำแหน่งที่กำหนดบนจอภาพด้วยคำสั่ง SAY นั้นในลักษณะ full-screen กล่าวคือ สามารถเลื่อน cursor กลับไปกลับมาได้จนกว่าจะกดคีย์ RETURN เมื่อใส่ในตัวแปรตัวสุดท้ายคือ MDraw ซึ่งก็จะออกจากโหมดการทำงานของ full screen นี้ไปทำงานคำสั่งอื่นต่อไป

ข้อสังเกต ขณะที่รับข้อมูลจากคีย์บอร์ดนี้จะเห็นว่าตัวแปรแต่ละตัวได้ถูกกำหนดความยาวของการรับข้อมูลลงในตัวแปรนั้น ๆ ได้โดยจะมีเครื่องหมายจุดคู่ (colon :) ปิดหัว ปิดท้ายของเนื้อที่ที่จะรับข้อมูลลงในตัวแปรนั้นด้วย ในที่นี้เรากำหนดด้วยคำสั่ง STORE นั่นเอง

ยิ่งกว่านั้นเรายังสามารถที่จะกำหนดรูปแบบของการรับข้อมูลให้รูปแบบที่เรียกว่า Format ที่สวยงามเหมาะสมกับความต้องการขึ้นไปอีกก็คือ การใช้คำสั่งที่มีรูปแบบทั่ว ๆ ไปดังต่อไปนี้

@ 〈coordinates〉 SAY ["prompt"] GET 〈variable〉 [PICTURE 〈format〉]

ให้สังเกตว่าเราจะมีการใช้คำสั่ง PICTURE แล้วตามด้วยรหัสกำหนดรูปแบบพิเศษ (Format) จะขอยกตัวอย่างเช่น

@ 5, 1 SAY "Today's date is" GET Date PICTURE "99/99/99"

ถ้าเราลองใช้คำสั่งดังกล่าวนี้แล้ว ดีเบสทูจะทำการแสดงผลของข้อความต่อไปนี้ในบรรทัดที่ 5 เริ่มจากคอลัมน์ที่ 1 ดังนี้

Today's date is : / / :

ในที่นี้สมมติว่าข้อมูลในตัวแปร Date เดิมมีค่าเป็นช่องว่างอยู่ยังไม่มีข้อมูลอย่างอื่น และในกรณีนี้เนื่องจากเรากำหนดรูปแบบดังกล่าวนั้น เราจะสามารถป้อนข้อมูลที่เป็นตัวเลขได้อย่างเดียวเท่านั้น

รูปแบบของรหัสกำหนดฟอร์มการรับข้อมูล หรือ FORMAT นี้ เราจะใช้รหัสต่าง ๆ ดังนี้

? หรือ	#	จะเป็นรหัสที่จะยอมให้ป้อนข้อมูลตัวเลขเท่านั้น
	A	จะรับข้อมูลเฉพาะตัวอักษร
	!	จะเท่ากับแปลงตัวอักษรตัวเล็กให้เป็นตัวอักษรตัวใหญ่โดยอัตโนมัติ
	X	รับข้อมูลได้ทุกชนิด
	$	จะแสดงผลเครื่องหมาย "$" บนจอภาพ
	*	จะแสดงผลเครื่องหมาย "*" บนจอภาพ

โดยรหัสต่าง ๆ เหล่านี้ 1 รหัสจะยอมรับข้อมูลจากคีย์บอร์ดได้ 1 ตัว ดังนั้นโดยวิธีการจัดรูปฟอร์มของ Format นี้ จะช่วยทำให้ระบบดีเบสทูของเรานี้มีประสิทธิภาพในการรับข้อมูลได้ยอดเยี่ยมทีเดียว

การใช้คำสั่งเพื่อจัดรูปฟอร์มการพิมพ์ (SET FORMAT TO PRINT, @...SAY...USING)

จากที่ได้กล่าวมาแล้วว่า ถ้าเราใช้คำสั่ง SET FORMAT TO PRINT หลังจากนั้นเมื่อเราใช้คำสั่งผลลัพธ์จะออกไปแสดงผลบนเครื่องพิมพ์ทันทีแทนที่จะไปแสดงบนจอภาพ ในกรณีนี้จะต้องไม่มีการใช้คำสั่ง GET, PICTURE หรือ READ เพราะเครื่องพิมพ์จะรับคำสั่งเหล่านี้ไม่ได้ ดังนั้นรูปฟอร์มของคำสั่ง @ ที่จะใช้เพื่อทำการส่งข้อความหรือผลลัพธ์บางอย่างไปพิมพ์ออกที่เครื่องพิมพ์จะเป็นรูปทั่ว ๆ ไปดังนี้

@ 〈coordinates〉 SAY〈ตัวแปร/นิพจน์/ข้อความในเครื่องหมายคำพูด〉 〈USING format〉

หลังเครื่องหมาย @ นี้จะเป็นชุดของตัวเลข 2 ตัวที่คั่นด้วยเครื่องหมายลูกน้ำ (comma) โดยตัวแรกแสดงตำแหน่งของบรรทัดที่จะพิมพ์ตัวเลขตัวที่สองแสดงตำแหน่งคอลัมน์เริ่มต้นของการพิมพ์นั่นเอง การกำหนดตำแหน่งของการพิมพ์โดยตัวเลข 2 ตัวนี้ จะต้องเป็นตัวเลขที่เรียงลำดับกันไปจากน้อยไปหา

มากเท่านั้น เช่น เราต้องพิมพ์บรรทัดที่ 7 ก่อนที่จะพิมพ์บรรทัดที่ 9, สั่งพิมพ์ข้อความในคอลัมน์ที่ 15 ก่อนที่จะใช้คำสั่งพิมพ์ในคอลัมน์ที่ 63 เป็นต้นเหตุผลที่ต้องกำหนดตัวเลขการสั่งพิมพ์ในคำสั่งเรียงลำดับกันนั้น เพราะขีดจำกัดของเครื่องพิมพ์โดยทั่วไปที่ไม่สามารถพิมพ์ข้อความกลับไปกลับมาหรือขึ้น ๆ ลงนั่นเอง

คำสั่งพิมพ์โดยใช้เครื่องหมาย @ นี้ เราอาจจะใช้คำสั่ง GET แล้วตามด้วยตัวแปร วางตำแหน่งคำสั่งนี้ไว้หลัง SAY ได้เหมือนการใช้งานทั่ว ๆ ไปได้ เพื่อที่จะส่งค่าของตัวแปรนั้นออกมาพิมพ์ และยิ่งกว่านั้นยังสามารถที่จะกำหนดรูปแบบการพิมพ์ที่เรียกว่า Format ได้โดยการใช้คำสั่ง USING และตามด้วยรหัสของการจัดฟอร์มได้อีกเช่นกัน รหัสหรือเครื่องหมายของการจัดฟอร์มที่จะนำมาวางในคำสั่งหลัง USING จะมีได้หลายรหัสดังนี้

เครื่องหมาย ? หรือ # จะแทนการพิมพ์ตัวเลขเท่านั้น

A จะแทนการพิมพ์ตัวอักษรเท่านั้น

X จะแทนการพิมพ์อักขระที่เป็นอักษร หรือตัวเลข หรือเครื่องหมายก็ได้

$ เครื่องหมายที่แสดงว่าจะพิมพ์ตัวเลข หรือจะพิมพ์ "$" ในตำแหน่งของตัวเลขศูนย์ เฉพาะที่เป็นศูนย์นำหน้า

* เครื่องหมายที่แสดงว่าจะพิมพ์ตัวเลข หรือจะพิมพ์ "*" ในตำแหน่งของตัวเลขศูนย์ เฉพาะที่เป็นศูนย์นำหน้า

เพื่อความเข้าใจลองดูตัวอย่าง ถ้าเราใช้คำสั่งว่า

```
@ 10, 50 SAY Hours * Rate USING '$$$$$$$.99'
```

แต่อย่าลืมว่าเราต้องใช้คำสั่งนี้หลังจากที่ใช้คำสั่งว่า SET FORMAT TO PRINT เพื่อจะส่งคำสั่งข้างบนนี้ออกไปที่เครื่องพิมพ์ ถ้าสมมติว่าตัวแปร Hours = 8 และตัวแปร Rate = 12.73 แล้ว เราจะได้ผลลัพธ์ของคำสั่งพิมพ์ข้างบนนี้ไปแสดงบนเครื่องพิมพ์ในบรรทัดที่ 10 โดยเริ่มพิมพ์ในคอลัมน์ที่ 50 ว่า $$$$ 101.84- นั่นเอง ลองพิจารณาตัวอย่างโปรแกรมดู

ตัวอย่างที่ 1

เนื่องจากความสามารถของโปรแกรมดีเบสทูนี้ค่อนข้างจะมีประสิทธิภาพที่ดีมากทีเดียว โดยที่มีชุดคำสั่งให้ใช้งานตามความต้องการจำนวนมาก ประกอบกับเทคนิคพิเศษอื่น ๆ ซึ่งจะใช้ในการสร้างโปรแกรมเพื่อจะประมวลผลไฟล์ข้อมูลของผู้ใช้ วิธีการที่ดีที่สุดในการที่จะเรียนรู้เทคนิคต่าง ๆ เหล่านี้ และเพื่อที่จะให้คุ้นเคยกับไฟล์คำสั่งที่จะนำไปใช้งานได้ก็คือ ลองดูตัวอย่างโปรแกรมแล้วคีย์เข้าไปในเครื่องไมโครคอมพิวเตอร์แล้วทดสอบตัวอย่างโปรแกรมต่าง ๆ ดูนั่นเอง ในที่นี้จึงขอยกตัวอย่างโปรแกรมง่าย ๆ ที่อาจจะเขียนขึ้นมาเก็บไว้เป็นไฟล์คำสั่ง (command file) เพียงส่วนหนึ่งเก็บไว้ในดิสค์ได้ โดยเมื่อเราต้องการจะป้อนข้อมูลซึ่งเป็น เดือน/วันที่/ปี ไปเก็บไว้เป็นตัวแปรเพื่อที่จะนำไปประมวลผลข้อมูลอย่างอื่นต่อไป แต่ก่อนอื่นอาจจะต้องมีการตรวจสอบดูว่าข้อมูลที่ป้อนนั้นถูกต้องหรือเป็นไปได้หรือไม่ ในที่นี้จะเป็นโปรแกรมรับข้อมูลแล้วตรวจสอบดูว่าเดือนเป็นตัวเลขในช่อง 1-12, วันที่ระหว่างเลข 1-31 และต้องเป็นปี 1983 หรือไม่ แล้วจึงจะผ่านต่อไปทำโปรแกรมประมวลผลที่ตามมา

ดังนั้นลองดูตัวอย่างต่อไปนี้

```
erase                    คำสั่งเคลียร์จอภาพ
```

```
set talk off                  ควบคุมไม่ให้มีการแสดงผลลัพธ์ของคำสั่งปกติ
store "      " to mdate       กำหนดความกว้างของตัวแปร MDate ไว้ 8 ตำแหน่ง
store T to nodate             ตัวแปร NoDate เก็บค่าทางลอจิกไว้เป็น "T" (true)
do while nodate = T           โปรแกรมจะทำคำสั่งในระหว่าง DO WHILE กับ ENDDO
                              ในขณะที่ NoDate = "T" เท่านั้น
      @ 5,5 say "Set date MM/DD/YY" get mdate picture;
      "99/99/99"
      read
      คำสั่งรับข้อมูลจากคีย์บอร์ดเก็บในตัวแปร MDate
      if val($(mdate,1,2))<1;
      .or. val($(mdate,1,2))>12;
      .or. val($(mdate,4,2))<1;
      .or. val($(mdate,4,2))>31;
      .or. val($(mdate,7,2))<>83
            store "........" to mdate
            @ 7,5 SAY "*** BAD DATE,PRESS RE-ENTER ***"
            store T to nodate
  else
      store F to nodate    เมื่อผู้ใช้ป้อนข้อมูลถูกต้องจะกำหนดให้

      NoDate = "F" (false)

      endif
      enddo  โปรแกรมจะมาถึงคำสั่ง ENDDO เมื่อผู้ใช้ป้อนข้อมูลเดือน/วันที่/ปี ถูกต้อง
      erase
```

เมื่อเราทำการเอ็กซิคิวต์โปรแกรมดังกล่าวนี้แล้ว จะได้ผลลัพธ์แสดงบนจอภาพดังนี้

Set date MM/DD/YY : / / :

และในขณะนี้คอมพิวเตอร์จะรอให้ผู้ใช้ป้อนข้อมูลที่เป็นตัวเลข 2 ตัวแรกเป็นเดือน 2 ตำแหน่งถัดมาเป็นวันที่ และเป็นปีอีก 2 ตัวเลขตามลำดับ หลังจากนั้นโปรแกรมในบรรทัดต่อมาจะทำการตรวจสอบดูว่าตัวเลขแสดงเดือนนั้นมีค่าอยู่ระหว่าง 1-12 หรือเปล่า ตรวจสอบดูว่าตัวเลข 2 หลักแสดงวันที่นั้นมีค่าระหว่าง 1-31 หรือไม่ และก็จะตรวจสอบด้วยว่าตัวเลขแสดงปีในที่นี้อนุญาตให้เป็นตัวเลข 83 เท่านั้น (เฉพาะปี 1983) คำสั่งในการตรวจสอบเงื่อนไขดังกล่าวนี้จะอาศัยคำสั่ง IF และฟังก์ชันพิเศษที่กล่าวถึงมาแล้วในบทนี้ดังนี้

ใช้ฟังก์ชัน $ (substring function) เพื่อที่จะดึงตัวอักษรเพียงบางตัวจากค่าของตัวแปรที่เก็บข้อมูลตัวอักษร (string variable) เพื่อจะดึงตัวอักษร 2 ตัวซึ่งแสดงเดือน วันที่และปี (เช่น จะดึงวันที่มาตรวจได้โดยการอ่านจากตัวแปร MDate ตั้งแต่ตำแหน่งที่ 4 โดยอ่านมาเพียง 2 ตัว ในที่นี้ใช้ฟังก์ชันว่า $(MDate, 4,2) เป็นต้น)

ใช้ฟังก์ชัน VAL เพื่อจะแปลงข้อมูลดังกล่าวตอนต้นซึ่งเก็บอยู่ในหน่วยความจำของเครื่องในลักษณะของอักขระแบบตัวอักษร ให้กลายเป็นการเก็บข้อมูลแบบตัวเลข เพื่อพร้อมที่จะนำไปคำนวณ หรือเปรียบเทียบแบบตัวเลขต่อไป เช่น เราใช้คำสั่งว่า VAL($MDate, 4, 2)) เป็นต้น

การใช้ไลจิกฟังก์ชันในการเปรียบเทียบข้อมูลที่หามาได้ว่าอยู่ในช่องที่ต้องการหรือเปล่า เช่น โดยจะตรวจสอบว่าวันที่เป็นตัวเลขระหว่าง 1-12 หรือไม่ เป็นต้น ในที่นี้เราใช้ไลจิกฟังก์ชัน .OR. (ถ้าเงื่อนไขอันใดอันหนึ่ง หรือทั้งสองเงื่อนไขเป็นจริง คำตอบของการเปรียบเทียบเงื่อนไขทั้งสองนี้ก็จะตอบว่า *จริง* ด้วย)

ในกรณีที่ค่าที่เราผู้ใช้ป้อนเข้าไปไว้ในตัวแปร MDate นั้นไม่ได้อยู่ในช่องค่าที่ถูกต้องแล้ว โปรแกรมจะกำหนดให้ค่าตัวแปร MDate กลายเป็น *ช่องว่าง* ใหม่ แล้วพิมพ์ข้อความซึ่งเป็นการบอกให้ผู้ใช้ป้อนข้อมูลใหม่โดยพิมพ์ข้อความว่า "**** BAD DATE, PLEASE RE-ENTER ****" แล้วจึงกลับไปทำงานตามคำสั่งรับข้อมูลใหม่นั่นเอง แต่ถ้าเราได้ป้อนข้อมูลเดือน/วันที่/ปี ให้มีค่าอยู่ในช่องที่กำหนดแล้วโปรแกรมจึงจะผ่านไปยังคำสั่งอื่นถัดไป (ในที่นี้จะทำงานไปที่คำสั่ง ERASE ล่างสุด... แล้วจึงทำคำสั่งต่อ ๆ ไปที่จะตามมา)

ตัวอย่างที่ 2

สมมติว่าเราจะสร้างตารางแสดงผลโดยใช้ฟิลด์ DEALER, ADDRESS และ PHONE จากไฟล์ DEALERS และฟิลด์ USRCOMPY, MACAHINTYPE และ DATEPUR จากไฟล์ WARRANTY เพื่อที่จะสร้างตารางแสดงผล เราจะสามารถใช้คำสั่ง @......SAY เพื่อสร้างตารางแสดงผลต่อไป

สมมติว่าเราจะสร้างตารางแสดงผลในรูปฟอร์มสดังนี้คือ

```
A>
                                   Dealer's progress report
                                   ************************

Dealer name : THAI PAIBOON
Address : 1/4 BANMAI MUENG DIST.
Changwat : CHONBURI




Customer list             Machine                Date
.............               ..                 ........
.............               ..                 ........
.............               ..                 ........
```

จะเห็นได้ว่า 3 ฟิลด์แรกได้มาจากไฟล์ DEALERS และ 3 ฟิลด์ถัดมาได้มาจาก WARRANTY ไฟล์เราจะใช้ชุดคำสั่งต่อไปข้างล่างสร้างโปรแกรมคำสั่ง (คอมมานด์ไฟล์) ที่ชื่อว่า *DHEADER*

```
SET FORMAT TO PRINT
EJECT
@ 1,28 SAY "DEALER'S PROGRESS REPORT"
@ 2,28 SAY "***............."
@ 4,6  SAY "DEALER NAME :"
@ 4,20 SAY P.DEALER
@ 5,10 SAY "ADDRESS :"
@ 5,20 SAY P.ADDRESS
@ 6,12 SAY "CHANGWAT"
@ 6,20 SAY P.CHANGWAT
@ 8,5  SAY "CUSTOMER LIST"
@ 8,46 SAY "MACHINE"
@ 8,61 SAY "DATE"
```

จากโปรแกรมส่วนที่กล่าวมานี้พออธิบายได้ดังนี้

1. SET FORMAT TO PRINT เมื่อใช้คำสั่งนี้ คำสั่ง @ จะนำข้อมูลออกไปที่เครื่องพิมพ์ คู่ลำดับ 0, 0 หมายถึง มุมบนซ้ายสุดของจอมอนิเตอร์ คำสั่ง @ เมื่อดีเบสทูกระทำจะได้ผลลัพธ์ออกมาอาจจะเป็นได้ว่าออกมาทางจอมอนิเตอร์หรือเครื่องพิมพ์ โดยถ้าเราใช้คำสั่ง SET FORMAT TO SCREEN ผลลัพธ์ทั้งหมดจะออกมาทางจอมอนิเตอร์แทนที่จะออกทางเครื่องพิมพ์

2. EJECT เป็นคำสั่งให้ผลลัพธ์ออกทางเครื่องทำร่วมกับคำสั่ง SET FORMAT TO PRINT ซึ่งคำสั่ง EJECT นี้จะกระทำการจัดให้รีจิสเตอร์แถวและคอลัมน์รีจิสเตอร์เป็นศูนย์

3. ตัวเลขที่กำหนด เช่น 1, 25 หมายถึง ให้เริ่มพิมพ์แถวที่ 1 คอลัมน์ที่ 25 ดีเบสทูจะทำการควบคุมการพิมพ์ตามแถวและคอลัมน์ดังกล่าว

ในการใช้หมายเลขแถวและคอลัมน์นี้ถ้าหากต้องการให้ออกทางเครื่องพิมพ์จำเป็นต้องให้ตัวเลขแถวเรียงกันมาตั้งแต่ 1 ไปทางเลขมากขึ้น เครื่องพิมพ์จะไม่สามารถที่จะพิมพ์ย้อนไปย้อนมาได้แต่ถ้าหากว่าการเซทรูปแบบนี้ออกทางจอภาพแล้ว การจัดแถวจะกลับไปมาได้

หากต้องการแก้ไขในโปรแกรมที่อยู่ในคอมมานด์ไฟล์เราทำได้โดยการพิมพ์ข้อความ MODIFY COMMAND DEALER ดีเบสทูจะนำเอาโปรแกรมเดิมที่เขียนไว้แล้วออกมาคราวนี้เราจะแก้เฉพาะ 3 บรรทัดแรก

```
USE WARRANTY
SET FORMAT TO SCREEN
ERASE
```

การเก็บคอมมานด์ไฟล์ไว้ในแผ่นจานแม่เหล็ก เรากระทำได้โดยกดคีย์ CTRL-W การเก็บคอมมานด์ไฟล์จะเก็บในชื่อตามที่บอกไว้ก่อนแล้วโดยมีชนิดไฟล์เป็น .CMD หากต้องการเรียกคอมมานด์ไฟล์นี้มาใช้ก็เพียงแต่ใช้คำสั่ง DO ดังนี้

DO DEALER ดีเบสทูจะกระทำตามคำสั่งที่เก็บไว้ในคอมมานด์ไฟล์

คำสั่งเกี่ยวกับการจัดรายงานเพิ่มเติม

ในรูปแบบการพิมพ์นี้เรากำหนดไว้แล้วดังข้างต้นซึ่งเราไม่ทราบว่าช่องว่างในรูปแบบที่กำหนดการพิมพ์ของเราจะยาวเพียงพอสำหรับตัวแปรที่จะนำมาพิมพ์หรือไม่ ถ้าเราพิมพ์ฟิลด์หนึ่งยาวล้ำไปยังอีกฟิลด์หนึ่งก็จะเกิดปัญหาการพิมพ์ในฟิลด์ถัดไป

เพื่อที่จะทำการแก้ที่เราพิมพ์ผลลัพธ์โดยคำสั่ง @...SAY ".....'' ลองพิจารณาจากการเขียนคำสั่งต่อไปนี้

```
. MODIFY COMMAND REPORT
```

แล้วเปลี่ยนข้อความดังต่อไปนี้

```
@ ROWNUM,11  SAY  S.USRCOMPY
@ ROWNUM,40  SAY  S.MACHINTYPE
@ ROWNUM,61  SAY  S.DATEPUR
```

โดยแก้เป็น

```
@ ROWNUM,11  SAY  S.USRCOMPY USING " ****************** "
@ ROWNUM,46  SAY  S.MACHINTYPE USING  " ******* "
@ ROWNUM,61  SAY  S.DATEPUR USING " ******** "
```

การใช้ USING จะกำหนดรูปแบบที่พิมพ์ออกมาเป็นรูปแบบการพิมพ์เป็นแบบต่อเนื่อง (อนุกรม) ของตัวอักษรที่แสดงถึงตัวหนังสือที่ออกมาบนจอภาพ และจากที่แสดงดังตัวอย่าง S. USRCOMPY, S. MACHINTYPE มีขนาด 25 และ 7 ตัวอักษร ตามลำดับ

ถ้าหาก USING สั้นกว่าสิ่งที่จะพิมพ์ส่วนที่เกินกว่าจะได้รับการตัดออก

ตัวอักษรฟอร์แมท	ฟังก์ชัน SAY	ฟังก์ชัน GET
?	สำหรับตัวเลขถัดไป	ยอมรับตัวเลขและ " ","+","-","."
X	สำหรับตัวอักษรถัดไป	ยอมรับตัวอักษรใด ๆ รวมทั้งตัวเลข
A	สำหรับตัวอักษรถัดไป	ยอมรับตัวอักษรอย่างเดียวเท่านั้น

รูปแบบการพิมพ์ที่มีประสิทธิภาพมากขึ้น

เพื่อทำความเข้าใจได้ดียิ่งขึ้น ลองพิจารณาคำสั่งต่อไปนี้

```
USE WARRANTY
APPEND
```

เรคอร์ดถัดไปจะแสดงออกมาพร้อมเลขที่เรคอร์ด และรูปฟอร์มของฟิลด์ หลังจากได้ใส่ข้อมูลในฟิลด์ของเรคอร์ดเสร็จแล้ว เรคอร์ดต่อไปจะปรากฏบนจอจนกระทั่งเรากด CTRL-W ก็จะเก็บเข้าในไฟล์

คราวนี้เราจะมาทำการสร้างรูปแบบการป้อนข้อมูลที่แตกต่างจาก APPEND ธรรมดาโดยการใช้คอมมานด์ไฟล์

```
USE WARRANTY
SET FORMAT TO SCREEN
STORE "Y" TO CONT
DO WHILE CONT "Y"
   ERASE
   APPEND BLANK
   GO BOTTOM
   @ 3,10 SAY "CARD NUMBER : "GET CARDNUM PICTURE "*****
   @ 4,10 SAY "SERIAL NUMBER FOR DISPLAY UNIT :";
     GET DAERIALNUM PICTURE "*****"
   @ 5,14 SAY "SERIAL NUMBER FOR PRINTER :";
     GET PSERIALNUM PICTURE "*****"
   @ 6,13 SAY "SERIAL NUMBER FOR KEY BOARD :"
     GET KSERIALNUM PICTURE "*****"
   @ 8,12 SAY "MACHINTYPE :";
     GET MACHINETYPE PICTURE "*******"
   @ 10,12 SAY "USER´S COMPY NAME :";
     GET USRCOMPY PICTURE "***......."
   @ 12,12 SAY "DEALER´S NAME :";
     GET DEALER PICTURE "*****......."
   @ 14,12 SAY "DATE PURCHASED :";
     GET DATEPUR PICTURE "99/99/99"
   READ
   @ 20,10 SAY "WANT TO CONTINCE?" GET CONT
   READ
ENDDO
```

พิจารณาความหมายของโปรแกรมนี้อย่างละเอียดดู จะเห็นว่าคอมมานด์ไฟล์ชื่อ WAPPEND นี้มีส่วนช่วยเราในการป้อนข้อมูลเป็นอย่างมาก กล่าวคือ เราเริ่มด้วยคำสั่งบรรทัดแรกให้ใช้ไฟล์ชื่อ WARRANTY หลังจากนั้นก็ทำการเซทฟอร์แมทออกทางจอภาพ แล้วทำการเซทค่า CONT ซึ่งเป็นตัวแปรให้เป็นลอจิก Y (จริง)

เมื่อเริ่มคำสั่ง DO เราจะมีเงื่อนไขว่าทำจนกว่า CONT จะเป็นเท็จ (N,F) คำสั่ง ERASE จะทำการลบจอภาพ APPEND BLANK คือการเรียกเรคอร์ดว่าง ๆ สุดท้ายออกมา แล้วไปยังเรคอร์ดนี้ด้วยการใช้คำสั่ง GO BOTTOM

คำสั่งในกลุ่ม SAY และ GET นี้ ส่วนของ SAY จะแสดงผลบนจอภาพเป็นรูปแบบตามแถวและคอลัมน์ที่กำหนดส่วน GET จะร่วมกับ READ ทำการอ่านข้อมูลที่เราคีย์เข้า การใช้ GET หลายอันนี้จะทำให้เราคีย์ข้อมูลได้เหมือนเป็นการป้อนโดยตรง เมื่อป้อนจบทุกฟิลด์แล้วดีเบสทูจึงค่อยอ่านทั้งหมดไป การอ่านจะอ่านครั้งละเรคอร์ดหรือทุกฟิลด์เลย

หากระหว่างการพิมพ์มีข้อมูลที่ผิดพลาดสามารถแก้ไขได้โดยการเลื่อนตำแหน่งเคอร์เซอร์ไปแก้ไขตำแหน่งการควบคุมเคอร์เซอร์เป็นคีย์ที่ได้เคยกล่าวมาแล้ว จะเห็นว่าการป้อนข้อมูลแบบนี้จะมีรูปแบบที่สวยงามกว่าการใช้ APPEND อย่างเดียว เราสามารถใช้การจัดให้เป็นรูปฟอร์ม เช่น ใบเสร็จ อินวอย ใบส่งสินค้าได้เลย

ส่วนคำสั่ง "WANT TO CONTINUE ?" GET CONT ถ้าหากตอบ Y ก็จะทำการป้อนเรคอร์ด ต่อไปถ้าไม่ต้องการป้อนเรคอร์ดต่อไปก็คีย์ N

บทสรุป

ที่กล่าวมานี้เป็นเพียงตัวอย่างส่วนหนึ่งเท่านั้น ในการใช้งานจริงอาจจำเป็นจะต้องใช้ฟังก์ชันและคำสั่งพิเศษอื่น ๆ ที่กล่าวมาแล้วนั้น เพื่อที่จะทำให้โปรแกรมที่เราเขียนขึ้นมีความวิจิตรพิสดารยิ่งขึ้น เราก็จะขอกล่าวถึงและยกตัวอย่างโปรแกรมให้มากขึ้นในบทต่อ ๆ ไป ในบทนี้ก็เพียงแต่พยายามอธิบายหลักการทำงานของแต่ละคำสั่งซึ่งมีอยู่มากมาย ผู้ใช้*ไม่จำเป็น* จะต้องพะวงว่าจะมานั่งท่องจำกันทุกคำสั่ง เพียงแต่ให้ผ่านสายตาไปบ้าง เมื่อต้องการจะใช้งานจริง หรือเมื่อจำรายละเอียดไม่ได้ค่อยพลิกกลับมาอ่านดูนั่นเอง

การออกแบบระบบฐานข้อมูล

จากที่ได้เคยกล่าวมาแล้วว่าการกำหนดไฟล์นั้น โดยปกติจะเป็นไฟล์ชนิดลำดับที่มีการค้นหาข้อมูลจะกระทำในลักษณะเรียงกันไปทีละเรคอร์ดซึ่งก็ต้องเสียเวลามากเมื่อมีข้อมูลในไฟล์มาก เพื่อทำให้เวลาในการหาข้อมูลสั้นเข้าจึงจำเป็นต้องมีการจัดเรียงเรคอร์ดและใช้ไฟล์ในรูปแบบอินเด้กซ์ไฟล์ ลักษณะการค้นหาข้อมูลจะกระทำได้รวดเร็วมากขึ้น และโครงสร้างของระบบไฟล์ก็ง่ายต่อการใช้งานขึ้น รูปแบบของการสร้างอินเด็กซ์ไฟล์เป็นดังนี้

INDEX ON<ชื่อฟิลด์ที่เป็นคีย์> TO <ชื่อไฟล์ใหม่ที่เป็นอินเด็กซ์ไฟล์>

ในการสร้างอินเด็กซ์ไฟล์นี้จำเป็นต้องมีคีย์ฟิลด์ โดยการสร้างจะได้ไฟล์ใหม่โดยเป็นชนิด .NDX และจะแสดงให้เห็นได้ขณะใช้ DISPLAY FILE หรือ DIR ในซีพีเอ็ม ลักษณะของอินเด็กซ์ไฟล์นี้จะทำการจัดเรียงลำดับของคีย์ฟิลด์ไว้ การใช้อินเด็กซ์ไฟล์จะใช้ได้ต่อไปเมื่อได้ทำการ USE ไฟล์แม่ของอินเด็กซ์ไฟล์แล้วเท่านั้น การเก็บไฟล์อินเด็กซ์มิได้ทำให้ใช้พื้นที่เพิ่มเป็น 2 เท่าของไฟล์แม่เลย ทั้งนี้เพราะในตัวไฟล์อินเด็กซ์นั้นจะเก็บในลักษณะเฉพาะตัวชี้กับคีย์ฟิลด์เท่านั้น

ในการสร้างระบบฐานข้อมูลนั้นจำเป็นต้องสร้างเอ็นทีทีของข้อมูลต่าง ๆ ขึ้น นั่นก็หมายถึงมีส่วนของไฟล์ต่าง ๆ เกิดขึ้นมากมาย ไฟล์บางไฟล์ก็มีความสัมพันธ์เกี่ยวข้องกันที่จะใช้ในการสร้างรายงานของระบบฐานข้อมูล ดังนั้นในการใช้งานฐานข้อมูลบ่อยครั้งที่ต้องใช้อินเด็กซ์ไฟล์มากกว่าหนึ่งไฟล์ดังเช่นจากฐานข้อมูลไฟล์ชื่อ WARRANTY โดยมีฟิลด์ต่าง ๆ หลายฟิลด์ ผู้สร้างไฟล์ชนิดอินเด็กซ์ไฟล์ อาจสร้างอินเด็กซ์คีย์ไว้หลาย ๆ คีย์ก็ได้เช่นจากโครงสร้างอาจกำหนดคีย์ฟิลด์เป็น CARDNUM DSERIALNUM PSERIALNUM และ KSERIALNUM ซึ่งถ้าพิจารณาอย่างละเอียดจะเห็นว่าในกรณีที่มีการเปลี่ยนแปลงข้อมูลใด ๆ เกิดขึ้นไม่ว่าขณะใด เวลาใด จะสร้างปัญหาต่อข้อมูลได้ ดังนั้นในการแก้ไขข้อมูลในอินเด็กซ์ไฟล์ตัวใดตัวหนึ่งก็จะต้องแก้ไขให้ได้ทั้งหมดทุกไฟล์ทางแก้ไขในดีเบสทูคือการใช้คำสั่ง SET ซึ่งมีรูปแบบของคำสั่งคือ

SET INDEX TO <INDEX FILES LIST>

ส่วนของชื่อไฟล์อินเด็กซ์ทั้งหมดจะยอมได้จำนวน 7 ไฟล์ ชื่อของไฟล์แรกในส่วนไฟล์ที่เป็นชื่อเรียงกันนั้นถือว่าเป็นมาสเตอร์อินเด็กซ์ ในการหาค่าต่าง ๆ หรือกระทำในข้อมูลอาจจะทำเพียงไฟล์นี้ก็พอเพียงแล้วจากนั้นดีเบสทูจะทำงานตามคำสั่งของมาสเตอร์อินเด็กซ์

ลองมาพิจารณาดูว่าการกระทำเกี่ยวกับอินเด็กซ์ไฟล์ในฐานข้อมูลไฟล์เดียวกันนี้จะเป็นอย่างไร โดยลองใช้คำสั่งดังโปรแกรมต่อไปนี้

```
.
.
. USE WARRANTY
. INDEX ON CARDNUM TO CINX
00013 RECORDS INDEXED
. INDEX ON DSERIALNUM TO DINX
00013 RECORDS INDEXED
. INDEX ON PSERIALNUM TO PINX
00013 RECORDS INDEXED
. INDEX ON KSERIALNUM TO KINX
00013 RECORDS INDEXED
. USE WARRANTY INDEX CINX
. SET INDEX TO CINX,DINX,PINX,KINX
. APPEND
```

การสร้างไฟล์ WARRANTY โดยให้มาสเตอร์อินเด็กซ์คือCINX NDX นี้สามารถกำหนดอินเด็กซ์คีย์อื่น ๆ อีกได้ การป้อนข้อมูลในไฟล์มาสเตอร์อินเด็กซ์จะมีผลต่อไฟล์อื่น ๆ ทั้งหมดได้โดยตรง ทั้งนี้เพราะเราได้กระทำการเซทค่าอินเด็กซ์ไว้แล้วในคำสั่ง SET

เพื่อให้แน่ใจและเข้าใจยิ่งขึ้นลองทำการป้อนข้อมูลต่อไปนี้ลงไปหลังจากเสร็จสิ้นคำสั่ง APPEND แล้ว

```
00014  1111  1111  1111  1111  DU    SAHAVIRIYA LTD.  THAISAMUTH    12/08/83
00015  2222  2222  2222  2222  KU    ROONGRUENGSIN    TANAKIT       09/09/83
```

เมื่อเสร็จแล้วกด CTRL-W เพื่อทำการเก็บข้อมูลทั้งหมดลงในแผ่นจานแม่เหล็ก ลองพิจารณาด้วยการใช้คำสั่ง LIST ในส่วนของอินเด็กซ์ไฟล์แต่ละตัวว่าอยู่ในลำดับเรียงข้อมูลแต่ละเรคอร์ดถูกต้องหรือไม่ โดยใช้คำสั่งดังโปรแกรมต่อไปนี้

```
. USE WARRANTY INDEX CINX
. LIST
00014   1111  1111  1111  1111 DU   SAHAVIRIYA LTD.  THAISAMUTH      12/08/83
00015   2222  2222  2222  2222 KU   ROONGRUENGSIN    TANAKIT         09/08/83
00005  23116 49371  3670 40215 MU   NANAPAN          MITTHAI INSURE  10/07/83
00004  24115 48731 39511 40318 DU   SAHASILP         GENERAL EQUIP.  02/05/83
00013  24528 40531 32031 41112 MI   BANYAI           N.S.D.L.P.      08/08/83
00008  24532 42138 32025 43251 DU   TAE KUAN         RANGSIT         13/05/83
00003  25041 46715 38921 49116 KU   SRISAWANG        THAI PAIBOON    08/11/83
00002  25056 43815 36241 46166 DU   SRISAWANG        RANGSIT         05/09/83
00001  25163 41825 36415 43152 KU   VIMOLSIL         THAI PAIBOON    02/12/83
```

```
00010   25316 44567 32053 48721 MT      SIAMPANICH        MITTHAI INSURE   12/09/83
00006   25416 49111 35201 41026 KT      BONSILPCHAI       HOUSE OF SIAM    23/08/83
00011   25438 49008 35026 46525 HK      SIRIYONT          RANGSIT          03/06/83
00007   26438 43252 33511 41152 DU      MAHACHAK          DELCO INSURE     25/05/83
00009   26531 41312 33028 41123 KU      THAI PLASTIC      THAI PAIBOON     11/11/83
00012   43251 48320 37092 43211 DU      NGAMCHAI          RANGSIT          03/09/83
.
.
.
.
.
.
. USE WARRANTY INDEX DINX
. LIST
00014    1111  1111  1111  1111 DU      SAHAVIRIYA LTD.   THAISAMUTH       12/08/83
00015    2222  2222  2222  2222 KU      ROONGRUENGSIN     TANAKIT          09/08/83
00013   24528 40531 32031 41112 MI      BANYAI            N.S.D.L.P.       08/08/83
00009   26531 41312 33028 41123 KU      THAI PLASTIC      THAI PAIBOON     11/11/83
00001   25163 41825 36415 43152 KU      VIMOLSIL          THAI PAIBOON     02/12/83
00008   24532 42138 32025 43251 DU      TAE KUAN          RANGSIT          13/05/83
00007   26438 43252 33511 41152 DU      MAHACHAK          DELCO INSURE     25/05/83
00002   25056 43815 36241 46166 DU      SRISAWANG         RANGSIT          05/09/83
00010   25316 44567 32053 48721 MT      SIAMPANICH        MITTHAI INSURE   12/09/83
00003   25041 46715 38921 49116 KU      SRISAWANG         THAI PAIBOON     08/11/83
00012   43251 48320 37092 43211 DU      NGAMCHAI          RANGSIT          03/09/83
00004   24115 48731 39511 40318 DU      SAHASILP          GENERAL EQUIP.   02/05/83
00011   25438 49008 35026 46525 HK      SIRIYONT          RANGSIT          03/06/83
00006   25416 49111 35201 41026 KT      BONSILPCHAI       HOUSE OF SIAM    23/08/83
00005   23116 49371  3670 40215 MU      NANAPAN           MITTHAI INSURE   10/07/83
.
.
.
.
. USE WARRANTY INDEX PINX
. LIST
00014    1111  1111  1111  1111 DU      SAHAVIRIYA LTD.   THAISAMUTH       12/08/83
00015    2222  2222  2222  2222 KU      ROONGRUENGSIN     TANAKIT          09/08/83
00005   23116 49371  3670 40215 MU      NANAPAN           MITTHAI INSURE   10/07/83
00008   24532 42138 32025 43251 DU      TAE KUAN          RANGSIT          13/05/83
00013   24528 40531 32031 41112 MI      BANYAI            N.S.D.L.P.       08/08/83
00010   25316 44567 32053 48721 MT      SIAMPANICH        MITTHAI INSURE   12/09/83
00009   26531 41312 33028 41123 KU      THAI PLASTIC      THAI PAIBOON     11/11/83
00007   26438 43252 33511 41152 DU      MAHACHAK          DELCO INSURE     25/05/83
00011   25438 49008 35026 46525 HK      SIRIYONT          RANGSIT          03/06/83
00006   25416 49111 35201 41026 KT      BONSILPCHAI       HOUSE OF SIAM    23/08/83
00002   25056 43815 36241 46166 DU      SRISAWANG         RANGSIT          05/09/83
00001   25163 41825 36415 43152 KU      VIMOLSIL          THAI PAIBOON     02/12/83
00012   43251 48320 37092 43211 DU      NGAMCHAI          RANGSIT          03/09/83
00003   25041 46715 38921 49116 KU      SRISAWANG         THAI PAIBOON     08/11/83
00004   24115 48731 39511 40318 DU      SAHASILP          GENERAL EQUIP.   02/05/83
. USE ____
.
.
.
.
. USE WARRANTY INDEX KINX
. LIST
00014    1111  1111  1111  1111 DU      SAHAVIRIYA LTD.   THAISAMUTH       12/08/83
00015    2222  2222  2222  2222 KU      ROONGRUENGSIN     TANAKIT          09/08/83
00005   23116 49371  3670 40215 MU      NANAPAN           MITTHAI INSURE   10/07/83
00004   24115 48731 39511 40318 DU      SAHASILP          GENERAL EQUIP.   02/05/83
00006   25416 49111 35201 41026 KT      BONSILPCHAI       HOUSE OF SIAM    23/08/83
00013   24528 40531 32031 41112 MI      BANYAI            N.S.D.L.P.       08/08/83
00009   26531 41312 33028 41123 KU      THAI PLASTIC      THAI PAIBOON     11/11/83
00007   26438 43252 33511 41152 DU      MAHACHAK          DELCO INSURE     25/05/83
00001   25163 41825 36415 43152 KU      VIMOLSIL          THAI PAIBOON     02/12/83
00012   43251 48320 37092 43211 DU      NGAMCHAI          RANGSIT          03/09/83
00008   24532 42138 32025 43251 DU      TAE KUAN          RANGSIT          13/05/83
00002   25056 43815 36241 46166 DU      SRISAWANG         RANGSIT          05/09/83
00011   25438 49008 35026 46525 HK      SIRIYONT          RANGSIT          03/06/83
```

```
00010 25316 44567 32053 48721 MT        SIAMPANICH        MITTHAI INSURE  12/09/83
00003 25041 46715 38921 49116 KU        SRISAWANG         THAI PAIBOON    08/11/83
.
.
```

การป้อนข้อมูลในไฟล์มาสเตอร์อินเด็กซ์เพียงไฟล์เดียวจะมีผลต่อไฟล์อินเด็กซ์อื่น ๆ ได้ครบหมด ซึ่งเป็นข้อดีสำหรับการแก้ไขในไฟล์เพื่อไม่ให้เกิดปัญหาในกรณีที่มีการอินเด็กซ์กันหลาย ๆ คีย์ฟิลด์

การอินเด็กซ์ด้วยคีย์ฟิลด์หลายคีย์

จากที่กล่าวมาแล้วทั้งหมดเป็นการประยุกต์การใช้อินเด็กซ์ไฟล์โดยการกำหนดคีย์ฟิลด์เพียงคีย์เดียวต่ออินเด็กซ์ไฟล์หนึ่งไฟล์ แต่ในงานที่ประยุกต์หลายครั้งผู้ใช้ต้องการกำหนดคีย์ฟิลด์มากกว่าหนึ่งคีย์ในอินเด็กซ์ไฟล์ ๆ เดียว เช่น สมมติว่าในไฟล์ฐานข้อมูลชื่อว่าไฟล์ DEALERS ซึ่งเป็นไฟล์ที่ทำการเก็บชื่อลูกค้า และผู้ใช้ต้องการกำหนดคีย์ฟิลด์เป็น COUNTRY และ AREA ในการเป็นคีย์เพื่อค้นหาข้อมูล และเพื่อเป็นไปตามจุดมุ่งหมายดังกล่าวแล้วก็อาจจะทำได้ด้วยดีเบสทูดังโปรแกรมต่อไปนี้

```
. USE DEALERS
. INDEX ON CHANGWAT+AREA TO REGION
00015 RECORDS INDEXED
. USE DEALERS INDEX REGION
. LIST OFF
SIAM PAIBOON      93/5 CHAREANMUENG              BANGKOK     A0001 L00   28
MITTHI INSURE     90/5 RAMBUTTREE CHAK           BANGKOK     A0001 L00   22
DELCO INSURE      346 TROK CHAN YANNAWA          BANGKOK     A0001 L00   15
N.S.D.L.P.        115 SIRICHINDA CHAI            BANGKOK     A0001 L01   20
CHUSENG HUAT L.   83/3 POONIMIT BUMRUNG          CHAENGMAI   A0002 L00   11
INTER-MARKETING   364 PRAHOLYOTIN PAYAO          CHENGRAI    A0002 L01   20
THAI PAIBOON      1/4 BANMAI MUENG DIST.         CHONBURI    A0001 L01   10
HOSE OF SIAM      28 MUKSALA DEEWAN              LAMPANG     A0002 L00   10
RANGSIT           15/8 SONGKRANVITEE             LOPBUREE    A0001 L01   12
SKULTHAI CO,LTD   155 SOI SENASARID              NONTABURI   A0001 L00    6
D.T.C. SUPPLY     11 RONGARUN WANGTONG           PITSANULOK  A0002 L01   13
GENERAL EQUIP.    525 PORNSAWANG SUKHUMVIT       RAYONG      A0003 L00   12
COMMERCIAL CO.    13/9 SUANYAI SAMPAYA           SONGKLA     A0004 L01    9
PAISAL SILP CO.   199 VISETKUL                   TRANG       A0004 L00   11
CITY'S SPARCK     16 UBONSAK MUENG DIST.         UBON        A0003 L01    8
.
.
.
```

ในการใช้อินเด็กซ์ไฟล์ที่มีคีย์ฟิลด์เพียงคีย์เดียว เมื่อเราต้องการค้นหาก็จะหาได้โดยใช้ชื่อคีย์นั้น เช่น

```
. USE DEALERS
. INDEX ON CHANGWAT TO DISTRIC
00015 RECORDS INDEXED
. USE DEALERS INDEX DISTRIC
. FIND BANGKOK
. DISPLAY
00002  SIAM PAIBOON    93/5 CHAREANMUENG      BANGKOK     A0001 L00   28
```

การค้นหาจะหาชื่อ THAILAND ในการใช้เป็นคีย์ในการค้นหาข้อมูลในเรคอร์ดนั้น แต่เมื่อใช้ 2 คีย์ฟิลด์ ในการเป็นอินเด็กซ์คีย์ เราอาจใช้คีย์ด้วยการรวมข้อความทั้งสองฟิลด์ได้ เช่น

```
. USE DEALERS INDEX REGION
. FIND CHAENGMAI A0002
. DISPLAY
00009  CHUSENG HUAT L. 83/3 POONIMIT BUMRUNG     CHAENGMAI  A0002 L00   11
```

หรืออาจจะเขียนให้สั้นกะทัดรัดก็ได้ โดยใช้ชื่อย่อคีย์ เช่น

```
. FIND CHAENGMAI A
. DISPLAY
00009  CHUSENG HUAT L. 83/3 POONIMIT BUMRUNG     CHAENGMAI  A0002 L00   11
```

ในการใช้หลายฟิลด์เป็นคีย์นั้นในการใช้คำสั่ง FIND นั้นค่าที่ต้องการค้นหาในฟิลด์แรกเป็นตัวที่ควรเขียนให้ครบ แต่ในฟิลด์ถัดไปจะใช้ตัวย่อก็ได้เพื่อจะได้ไม่เกิดปัญหาในการค้นหา เพราะถ้าใช้ตัวย่อในฟิลด์แรกแล้วอาจจะได้เรคอร์ดที่ตรงที่ค้นหาแต่เป็นเรคอร์ดที่ไม่ต้องการก็ได้

การใช้อินเด็กซ์ไฟล์เป็นเรื่องที่น่าสนใจและควรนำมาใช้ในการค้นหาข้อมูลมาก ทั้งนี้เพราะการใช้งานคำสั่งอินเด็กซ์ เช่น FIND นั้นสามารถใช้ได้ง่าย

สำหรับคำสั่ง SORT นั้นเป็นเรื่องที่น่าทำความเข้าใจโดยเปรียบเทียบกับ INDEX เพราะมีลักษณะการจัดเรียงเหมือนกัน แต่การใช้ SORT นั้นจะมีการสร้างไฟล์ใหม่ที่ได้รับการจัดเรียงแล้วให้มีขนาดใหญ่เท่ากับไฟล์เดิม ดังนั้นถ้าไฟล์เดิมตัวโตจะทำให้สิ้นเปลืองเนื้อที่ในแผ่นจานแม่เหล็กมากซึ่งเป็นเรื่องที่ไม่น่าใช้ แต่ถ้าใช้ INDEX ส่วนที่เพิ่มเติมเนื้อที่แผ่นจานแม่เหล็กในส่วน .NDX ไฟล์จะมีขนาดไม่ใหญ่โตมากนัก จึงน่าใช้มากกว่าคำสั่ง SORT

การทำข้อมูลในไฟล์ให้ใหม่เสมอ (UPDATE)

ในการแก้ไขข้อมูลในไฟล์ โดยอาจนำข้อมูลในอีกไฟล์หนึ่งเข้ามาแทนที่โดยการใช้ REPLACE หรือ ถ้าเป็นตัวเลขก็อาจใช้ SUM ก็ได้ ลองพิจารณาจากรูปแบบคำสั่งดังโปรแกรมต่อไปนี้

```
USE UPMASTER
LIST STRUCTURE
USE UPFROM
LIST STRUCTURE
GO TOP
DO WHILE .NOT.EOF
   SELECT SECONDARY
   USE UPMASTER
   GOTOP
```

```
    REPLACE ALL S.D WITH P.B FOR S.A=P.A
    SELECT PRIMARY
    SKIP
ENDDO
```

จากโปรแกรมที่เขียนนี้เรามีไฟล์ 2 ไฟล์ ไฟล์แรกชื่อ UPMASTER และไฟล์ที่สองชื่อ UPFORM ในการทำงานนี้เราให้ไฟล์ UPFORM เป็นไฟล์ปฐมภูมิ โดยในรูปการทำงานเราจะนำเอาไฟล์ใน UPFORM มาทีละเรคอร์ด แล้วตรวจสอบดูโดยเปิดไฟล์ทุติยภูมิ UPMASTER และตรวจสอบฟิลด์ A ว่าในส่วนของไฟล์ทุติยภูมิเรคอร์ดใดบ้างที่มีค่าตรงกับฟิลด์ A ในไฟล์ปฐมภูมิ ถ้าตรงกันจะนำค่าในฟิลด์ B ของไฟล์ปฐมภูมิมาแทนที่ในฟิลด์ D ของไฟล์ทุติยภูมิ ทำงานเช่นนี้จนหมด

การหาค่าผลรวมในฟิลด์ใดฟิลด์หนึ่ง

เราสามารถใช้คำสั่ง TOTAL มาทำการคำนวณหาค่าผลรวมของฟิลด์ใดฟิลด์หนึ่งที่เราต้องการหาและเก็บไว้ในฐานข้อมูลได้ ดังตัวอย่างเช่น

```
USE UPMASTER
SORT ON A TO SMASTER
USE SMASTER
TOTAL ON A TO SUMA1 FIELDS B,C
USE SUMA1
LIST STRUCTURE
LIST OFF
```

วิธีการดังกล่าวนี้ยังไม่ดีเพราะไฟล์ที่ได้ใหม่นี้จะมีฟิลด์อื่นที่ไม่ต้องการได้รับการลอกมาด้วย ในกรณีนี้อาจดัดแปลงใหม่ได้เป็น

```
COPY TO SUMA2 FIELD B,C STRUCTURE
USE SUMA2
LIST STRUCTURE
USE SMASTER
TOTAL ON ATO SUMA1 FIELDS B,C
USE SUMA2
LIST STURCTURE
LIST OFF
```

คราวนี้การทำงานของคำสั่ง TOTAL จะเป็นไปตามที่ต้องการ

การรวมสองไฟล์เข้าด้วยกัน

สมมติว่าเราต้องการสร้างไฟล์สำหรับเก็บข้อมูลลูกค้า ซึ่งใช้ชื่อไฟล์ว่า CUSTOMER โดยมี

ฟิลด์ทั้งสิ้น 3 ฟิลด์คือ USRCOMPY, COUNTRY, AREA โดยปกติลูกค้าจะอยู่ตามที่ต่าง ๆ และถ้าเราสมมติว่าลูกค้าแต่ละคนจะซื้อของจากตัวแทนร้านค้าในถิ่นนั้นไม่ซื้อจากที่อื่นอีกเลย ซึ่งเรามี DEALER FILE โดยสร้างจากฟิลด์ COUNTRY และ AREA จะอยู่ใน DEALERS นี้ด้วย ส่วนชื่อลูกค้าหรือฟิลด์ USRCOMRY จะอยู่ในไฟล์ WARRANTY โดยทั้งไฟล์ WARRANTY และ DEALERS จะมีฟิลด์ชื่อ DEALER ด้วยกัน เราจะทำการตรวจสอบในไฟล์ DEALERS ซึ่งเป็นไฟล์ปฐมภูมิว่าข้อมูลในฟิลด์ DEALER ตรงกับข้อมูล DEALER ในไฟล์ WARRANTY เราจะมาทำการสร้างไฟล์ใหม่ชื่อ CUST โดยไฟล์ CUST จะมีฟิลด์ 3 ฟิลด์คือ USRCOMPY, COUNTRY, AREA ลองดูคำสั่งดังโปรแกรมดังต่อไปนี้

```
. USE DEALERS
. SELECT SECONDARY
. USE WARRANTY
. SELECT PRIMARY
. JOIN TO CUST FOR $(P.DEALER,1,10)=$(S.DEALER,1,10);
        FIELD USRCOMPY,CHANGWAT,AREA
. USE CUST
. LIST STRUCTURE
STRUCTURE FOR FILE:   CUST.DBF
NUMBER OF RECORDS:    00010
DATE OF LAST UPDATE: I4/I5/I5
PRIMARY USE DATABASE

FLD        NAME        TYPE WIDTH    DEC
001      USRCOMPY       C     015
002      CHANGWAT       C     010
003      AREA           C     005

** TOTAL **                 00031
.
.
. LIST ALL OFF
VIMOLSIL          CHONBURI    A0001
SRISAWANG         CHONBURI    A0001
THAI PLASTIC      CHONBURI    A0001
SAHASILP          RAYONG      A0003
MAHACHAK          BANGKOK     A0001
BANYAI            BANGKOK     A0001
SRISAWANG         LOPBUREE    A0001
TAE KUAN          LOPBUREE    A0001
SIRIYONT          LOPBUREE    A0001
NGAMCHAI          LOPBUREE    A0001
```

จะเห็นได้ว่าไฟล์ CUST เกิดขึ้นใหม่โดยเป็นผลการรวมของไฟล์ DEALERS กับไฟล์ WARRANTY โดยฟิลด์ทั้งสามของไฟล์ CUST มาจากไฟล์ DEALERS กับ WARRANTY ข้อมูลที่จะสร้างมาเชื่อมถึงกันจะมีเงื่อนไขคือข้อมูลใน DEALER จำนวน 10 ตัวอักษรแรกต้องตรงกันนั่นเอง

การออกแบบระบบงานด้วยดีเบสทู

เมื่อผ่านมาจนถึงขั้นนี้แล้วเราคงพอเห็นแล้วว่าการจะสั่งงานให้ดีเบสทูทำงานให้เราได้นั้นต้องใช้ชุดคำสั่ง ชุดคำสั่งดังกล่าวเมื่อนำมารวมกันก็จะปรากฏออกเป็นโปรแกรมได้ การเรียงคำสั่งต่าง ๆ เข้าด้วยกันนี้จำเป็นต้องมีเทคนิคและวิธีการอยู่บ้าง แต่อย่างไรก็ตามรูปแบบการเขียนสั่งงานดีเบสทูเป็นภาษาที่ใกล้เคียงกับภาษาอังกฤษที่ใช้งานทั่วไปมาก และรูปโครงของคำสั่งก็สามารถจัดรูปแบบให้เป็นรูปแบบโปรแกรมโครงสร้างได้ ซึ่งก็นับว่าเป็นข้อดีเด่นประการหนึ่งของดีเบสทูนั่นเอง

แต่อย่างไรก็ตามการที่จะออกแบบฐานข้อมูลที่ยิ่งยากซับซ้อนนั้นไม่ใช่งานที่ทำได้ง่ายนักต้องอาศัยประสบการณ์และความชำนาญพอสมควร ซึ่งบทนี้จะกล่าวถึงการสร้างโปรแกรมซึ่งเมื่อเข้าใจได้แล้วก็จะเขียนโปรแกรมสั่งงานดีเบสทูได้ วิธีการเขียนยังต้องอาศัยเทคนิคและวิธีการซึ่งผู้เขียนอาจเขียนโปรแกรมได้แตกต่างกันออกไป

หลังจากที่เราได้คุ้นเคยกับไฟล์มาแล้วคือ ไฟล์ WARRANTY และไฟล์ DEALERS และ CUSTOMER ไฟล์ในบทนี้จะได้นำเอาไฟล์เหล่านี้มาใช้ในการออกแบบระบบงาน

จากโครงสร้างที่ได้เคยกล่าวมาในบทก่อน ๆ ที่ใช้เป็นตัวอย่างไฟล์ DEALERS มีโครงสร้างประกอบด้วยฟิลด์ต่าง ๆ คือ DEALER, ADDRESS CONTRY, AREA, CONTACT, PHONE, STAFFNUM ส่วนไฟล์ WARRANTY ประกอบด้วยฟิลด์ต่าง ๆ คือ CARDNUM, DSERIALNUM, PSERIALNUM, KSERIALNUM, MACHINTYPE, USRCOMPY, DEALER, DATEPUR ส่วนไฟล์ CUSTOMER เป็นไฟล์ที่สร้างขึ้นมาใหม่โดยเป็นไฟล์ที่รวม USRCOMPY ทั้งหมดของ WARRANTY ไฟล์ไว้นั่นเอง กล่าวคือ ทุกชื่อของ USRCOMPY จาก WARRANTY ไฟล์จะมาเก็บไว้ในไฟล์นี้ด้วย ดังนั้นหากมีการเพิ่ม USRCOMPY ให้กับ WARRANTY ไฟล์ก็จะต้องมาเพิ่มให้กับไฟล์ CUSTOMER นี้ด้วย

สิ่งที่ต้องการจากฐานข้อมูล

จากที่กล่าวมาแล้วว่าฐานข้อมูลที่ยกตัวอย่างนี้ประกอบด้วยไฟล์ 3 ไฟล์ และสิ่งที่เราต้องการจากฐานข้อมูลนี้ถ้าแยกแยะออกเป็นข้อตามความต้องการในการใช้งานพอแบ่งอยกได้เป็นดังนี้

ก. การเพิ่มข้อมูลในฐานข้อมูล ในกรณีนี้คือการเพิ่มเข้าไปยังไฟล์ WARRANTY สิ่งที่จะต้องตรวจจุดสำคัญสองแห่งคือ ถ้าไม่มี USRCOMPY ในไฟล์ CUSTOMER นั่นหมายความว่าเป็น USRCOMPY รายใหม่ ดังนั้นเราต้องเพิ่มเรคอร์ดเกี่ยวกับลูกค้ารายใหม่นี้ลงใน CUSTOMER ไฟล์ และอีกประการหนึ่งคือ ถ้าไม่มี DEALER ในส่วนข้อมูลที่อยู่ใน WARRANTY ที่ใช้งานอยู่นั้นก็แสดงว่าเราได้ข้อมูลผิด หรือจัดการบางอย่างผิด เพราะจะหมายความว่าไม่มีตัวแทนขายด้วย

ข. การถามหาข้อมูลจากฐานข้อมูล (ENQUIRY) สิ่งที่ต้องการในแง่ของผู้ใช้ก็คือ ถามหาข้อมูลที่เก็บไว้ในฐานข้อมูล โดยการถามหานั้นต้องเตรียมไว้ในรูปของโปรแกรม โดยเมื่อทำการรันโปรแกรม ผู้ใช้จะถามคำถามที่ต้องการแล้วกับดีเบสทูโดยคำถามที่ตั้งไว้ในกรณีนี้คือ คำถามเกี่ยวกับ DEALER โดยผู้ถามจะถามโดยการป้อนชื่อ DEALER แล้วดีเบสทูจะทำการตอบคำถามด้วยการแสดงที่อยู่ หมายเลขโทรศัพท์ contact ซึ่งเป็นข้อมูลเกี่ยวกับ DEALER นั้นนั่นเอง นอกจากนี้ผู้ใช้ยังสามารถถามข้อมูลเกี่ยวกับ CUSTOMER โดยเพียงป้อนข้อมูล CUSTOMER ดีเบสทูก็จะตอบข้อมูลเกี่ยวกับลูกค้าที่ผู้ใช้ต้องการทันที

ค. การรายงานผล บางครั้งเราอาจต้องการให้ดีเบสทูทำการรายงานผลโดยพิมพ์ออกทางเครื่องพิมพ์ตามรูปแบบและข้อมูลที่เราต้องการ

เพื่อเข้าใจในหลักการออกแบบระบบ ลองพิจารณาจากโปรแกรมต่อไปนี้ซึ่งเขียนขึ้นโดยอาศัยหลักฐานข้อมูลจากที่กล่าวแล้ว โดยจะเริ่มจากเมนโปรแกรมโดยเมนโปรแกรมจะแสดงเมนูให้ผู้ใช้โดยถ้าผู้ใช้เลือกเมนู (menu) ข้อใดก็จะกระทำตาม ในที่นี้จะใช้ชื่อโปรแกรมคอมมานด์นี้ว่า *TASK*

```
store t to cont
do while cont
store " " to choice
erase
@ 2,30 say chr(27)+chr(112) "***warranty system"+chr(27)+chr(113)
@ 5,25 say "1=append"
@ 7,25 say "2=enquiry"
@ 9,25 SAY "3=report"
@ 11,25 say "4=stop"
@ 15,25 say "please enter a number : " get choice
read
*control sequence
do case
     case choice="4"
          quit
     case choice="1"
          do tappend
     case choice="2"
          do tenquire
     case choice="3"
          do treport
     otherwise
          @ 20,20 say "**error, please enter again **"
endcase
```

หากใช้คำสั่ง MODIFY COMMAND เมื่อจบโปรแกรมจะต้องการเก็บข้อมูลลงในแผ่นจานแม่เหล็กเราจะต้องปิดท้ายด้วย CTL – W

สำหรับโปรแกรมที่เขียนนี้มีโครงสร้างที่ไม่ยุ่งยากอะไร เชื่อว่าจะทำความเข้าใจได้โดยไม่ยากเย็นสำหรับคำสั่งที่ใช้ CHR(27) + CHR(112) นี้ ในกรณีนี้จะบอกให้เครื่องแสดงตัวอักษรในรูปแบบรีเวิร์สวีดีโอ คือ ตัวอักษรดำบนพื้นขาว และ CHR(27 + CHR(113) เป็นการสั่งกลับมาแสดงผลแบบปกติ

จากโปรแกรมหลักที่ผ่านมานี้จะเห็นว่าโปรแกรมได้เรียกโปรแกรมย่อยอีกถึง 3 โปรแกรมด้วยกันคือ TAPPEND, TENQUIRY และ TREPORT ซึ่งจะทำหน้าที่เฉพาะงานที่แตกต่างกันออกไป

จะเห็นได้ว่าการที่จะให้เครื่องคอมพิวเตอร์กระทำตามโปรแกรมใด จะต้องมีการตรวจเงื่อนไขได้จากข้อมูลที่เราให้แก่มัน โดยการถามตามเมนูนั่นเอง

ในการตรวจสอบเงื่อนไขนั้นในที่นี้เราใช้ชุดคำสั่ง IF...ELSE...ENDIF ทำการตรวจสอบเงื่อนไขที่คีย์เข้า ถ้าถูกต้องตามเงื่อนไข เช่น เมื่อคีย์ 3 หรือ CHOICE = "3" ก็จะทำงานตามคำสั่งในโปรแกรมย่อย TREPORT และเมื่อทำใน TREPORT และพบกับคำสั่ง RETURN จึงกลับมายังโปรแกรมนี้ใหม่อีกครั้ง

การตรวจสอบเงื่อนไขนี้นอกจากจะใช้ IF...ELSE...ENDIF แล้ว หากมีเงื่อนไขหลาย ๆ เงื่อนไข ซึ่งสามารถเขียนด้วยคำสั่ง DO CASE ดังโปรแกรมต่อไปนี้

```
DO CASE
   CASE condition 1
        ............
   CASE condition 2
        ............
        .
        .
        .
   CASE CONDITION n
        ............
   OTHERWISE            จะมีหรือไม่มีก็ได้เช่นเดียวกับ ELSE
   .........
ENDCASE
```

จากโปรแกรมหลักที่แสดงเมนูหากเขียนใหม่โดยใช้ DO CASE สิ่งที่ได้จะเป็นดังนี้คือ

```
DO CASE
   CASE CHOICE ='4'
        .
        .
        .
ENDCASE
```

ในที่นี้ OTHERWISE หมายถึง กรณีอื่น ๆ นอกจาก 1 ถึง 4 นั่นเอง ซึ่งเราจะละไว้โดยไม่ต้องเขียนก็ได้

เมื่อได้โปรแกรมเมนูแล้วต่อไปนี้ก็จะเป็นโปรแกรมย่อยที่เรียกมาจากโปรแกรมเมนูที่เราจะต้องสร้างเพิ่มเติม

พิจารณาโปรแกรม TREPORT ซึ่งเป็นการสร้างรายงานตามที่ต้องการดูโปรแกรมต่อไปนี้

```
STORE T TO CONT
DO WHILE CNT
STORE ' ' TO NUM
ERASE
@ 2,30 SAY   'REPORT ROUTINES'
@ 5,25 SAY ' 1 DEALERS'  PROGRESS REPORT'
@ 7,25 SAY ' 2 WARRANTY  MASTER REPORT'
@ 9,25 SAY ' 3 CUSTOMERS'  REPORT'
@ 11,25 SAY' 4 STOP'
@ 15,25 SAY' PLEASE ENTER A NUMBER :' GET NUM
READ
DO CASE
   CASE NUM='4'
        RETURN
   CASE NUM='1'
        DO DPROGRESS
        RETURN
   CASE NUM='2'
        REPORT FORM REPTWARR
        RETURN
   CASE NUM='3'
        REPORT FORM REPTCUST
        RETURN
ENDCASE
```

ในกรณีนี้ถ้า NUM = '1' จะไปทำยังโปรแกรมย่อยอีกทีชื่อว่า DPROGRESS ในโปรแกรมย่อยนี้จะเป็นโปรแกรมจัดรูปแบบรายงาน เพื่อต้องการดูว่าตัวแทนจำหน่ายของเรานี้มีลูกค้าเพิ่มขึ้นอย่างไร โดยให้พิมพ์รายชื่อลูกค้าของตัวแทนจำหน่ายทั้งหมดออกมา ซึ่งการสร้างรายงานนี้ได้เคยกล่าวถึงแล้วในบทก่อน ๆ (ลองย้อนกลับไปดู) ซึ่งในที่นี้จะขอข้ามไปโดยไม่เขียนให้ดูใหม่

ส่วนกรณีที่ NUM = '2' ก็จะสั่งพิมพ์รายงานจากไฟล์รายงานที่เตรียมไว้ก่อนแล้วชื่อว่า REPTWARR หรือเอาข้อมูลในไฟล์ WARRANTY ทั้งหมดมาพิมพ์ซึ่งอาจจะมีการพิมพ์หัวกระดาษตามรูปแบบที่เคยกล่าวแล้วเช่นกัน

ส่วนกรณี NUM = '3' ก็เช่นเดียวกัน จะพิมพ์ข้อมูลทั้งหมดในไฟล์ CUSTOMER โดยเขียนในรูปแบบรายงาน

สำหรับการเรียกหาข้อมูลหรือการ ENQUIRY เราสามารถค้นหาข้อมูลในไฟล์ทั้ง 3 ไฟล์ คือ แต่ในที่นี้จะใช้คีย์ที่ค้นหาเพียงชื่อตัวแทนจำหน่าย ชื่อลูกค้า และวันที่

ลองมาทำความเข้าใจกับคอมมานด์ไฟล์ต่อไปนี้

```
store to cnt
do while cnt
store ' ' to num
erase
@ 2,30 SAY 'REPORT ROUTINES'
@ 5,25 SAY '1.DEALERS' PROGRESS REPORT'
@ 7,25 SAY '2.WARRANTY MASTER REPORT'
@ 9,25 SAY '3.CUSTOMERS' REPORT'
@ 11,25 SAY '4. STOP'
@ 15,25 SAY 'PLEASE ENTER A NUMBER:'GET NUM
READ
DO CASE
    CASE NUM='4'
         RETURN
    CASE NUM='1'
         DO PROGRESS
         RETURN
    CASE NUM='2'
         REPORT FORM REPTWARR
         RETURN
    CASE NUM='3'
         REPORT FORM REPTCUST
         RETURN
ENDCASE
```

ในการเรียกหาข้อมูลนี้นั้น จะประกอบด้วยการเรียกด้วยคีย์ 3 คีย์คือ

DEALER เมื่อดีเบสทูถามว่าชื่อตัวแทนจำหน่าย เราก็จะต้องพิมพ์ชื่อที่ต้องการลงไปและดีเบสจะตอบข้อมูลที่เกี่ยวกับตัวแทนจำหน่ายนั้นออกมาคือ แอดเดรส หมายเลขโทรศัพท์ และคอนแทรค

CUSTOMER เมื่อดีเบสถามชื่อผู้ใช้จะต้องคีย์ชื่อที่ต้องการ โปรแกรมจะตอบด้วยชื่อตัวแทนของลูกค้านั้น (จากไฟล์ WARRANTY) และแอดเดรส หมายเลขโทรศัพท์และคอนแทรค

วันที่ เมื่อป้อนวันที่โปรแกรมจะได้ค้นหาวันที่ที่ตรงกันในไฟล์ WARRANTY แล้วนำข้อมูลเรคอร์ดนั้นออกมา

ส่วนที่เหลือส่วนสุดท้ายคือ โปรแกรม TAPPEND เป็นการป้อนข้อมูลเพิ่มเติมโดยเมื่อป้อนข้อมูลแล้วก็จะเป็นผลทำให้ข้อมูลในฐานข้อมูลถูกต้องเสมอ ดูโปรแกรมต่อไปนี้

```
use dealers
index on dealer to ddinx
use customers
index on usrcompy to custinx
```

```
use warranty
set format to screen
store "y" to ct
do while ct="y"
    store "                    " to temp
    @ 3,10 say "dealers name: " get temp
    select secondary
    use dealer index ddinx
    find &temp
    store "  " to testnum
    if  testnum = 0
        select primary
        @ 4,10 say "dealer not registered"
        loop
     endif
     .
     .
     erase
     append
     .
     .
     select secondary
     use customer index custinx
     store $(p.usrcompy,1,10) to usr
     find &usr
     if ... = 0
     append to the customer data base
     .                                          ขึ้นกับผู้ใช้จะเขียนเพิ่ม
     .
     endif
     select primary
enddo
```

การเขียนโปรแกรมในการใช้งานมักจะแยกแยะออกเป็นโปรแกรมย่อย ซึ่งจะทำให้โปรแกรมเป็นโครงสร้างที่ง่ายขึ้น

ตัวอย่างการประยุกต์ระบบจัดการฐานข้อมูล

ระบบจัดการฐานข้อมูลเป็นระบบที่มีความสำคัญในทุกวงการ การมีโปรแกรมระบบจัดการฐานข้อมูลนั้นได้เข้ามามีส่วนทำให้ผู้ใช้ใช้งานได้ง่ายขึ้น แต่อย่างไรก็ตามตัวโปรแกรมดีเบสทูที่แพร่หลายนี้เป็นโปรแกรมที่จัดอยู่ในรูปแบบเสมือนเป็นภาษาคอมพิวเตอร์ภาษาหนึ่งที่สามารถนำมาเขียนเป็นโปรแกรมตามผู้ใช้ต้องการได้เทคนิคการเขียนโปรแกรมด้วยภาษาดีเบสนี้เป็นเทคนิคที่ควรศึกษา ทั้งนี้เป็นหลักการของระบบโปรแกรมโครงสร้างที่เป็นมาตรฐานในการเขียนและใช้งาน ซึ่งหลักการของโปรแกรมโครงสร้างนี้ทำให้การเขียนโปรแกรมที่ยุ่งยากได้อย่างมีระบบ ดังนั้นในระยะหลังนี้จึงมีผู้นิยมเขียนโปรแกรมในหลักการภาษาโครงสร้างกันมากขึ้น ในบทนี้จะเป็นการยกตัวอย่างการเขียนโปรแกรมด้วยภาษาดีเบส การเขียนจะเขียนในลักษณะที่ได้ทดลองกับเครื่องและเอาผลที่ได้ออกมาพิมพ์ให้ดู ซึ่งก็อยากให้เป็นตัวอย่างสำหรับการศึกษาและเขียนโปรแกรมประยุกต์ต่อไป

ตัวอย่างการประยุกต์สำหรับการทำสินค้าคงคลัง (inventory control) อย่างง่าย

สมมติว่าต้องการทำบัญชีสินค้าคงคลังอย่างง่ายโดยสมมติว่ามีไฟล์หลักซึ่งเก็บรายชื่อสินค้าทั้งหมด และจำนวนรายละเอียดเกี่ยวกับสินค้าไว้ ซึ่งในที่นี้จะเรียกว่ามาสเตอร์ไฟล์ ในส่วนของมาสเตอร์ไฟล์มีโครงสร้างดังนี้

โครงสร้างไฟล์

ชื่อไฟล์ master. dbf

ชื่อฟิลด์	ชนิด	ความยาว	อธิบาย
CODE	C	4	รหัสสินค้าเป็นตัวเลขหรืออักษร 4 ตัว
NAME	C	23	ชื่อสินค้าเป็นตัวอักษร 23 ตัว
COST	N	8, 2	ราคาสินค้าต่อหน่วย คิดทศนิยม 2 ตำแหน่ง
ONHAND	N	6	จำนวนสินค้าในคลังสินค้า
MIN	N	5	จำนวนสินค้าในคลังที่ต่ำสุดที่จะต้องเตือน

SALE	N	8	อัตราการขายสินค้า (จำนวนสินค้าที่ขาย/ระยะเวลา)
DATE	N	2	จำนวนวนที่คิดอัตราการขายสินค้า

แต่ในกิจการค้ามีการสั่งสินค้าเข้ามาและขายสินค้าออกไปทุกวัน ดังนั้นในแต่ละวันจะมีการบันทึกไว้ในรูปของการเป็นเรคอร์ดประจำวัน ในที่นี้ใช้ชื่อไฟล์ว่า DAILY โดยมีโครงสร้างของไฟล์เป็นดังนี้

ชื่อไฟล์ DAILY. DBF

ชื่อฟิลด์	ชนิด	ความยาว	หมายเหตุ
ITEM	N	4	ลำดับชิ้นสินค้าที่เข้ามา
TRAN	C	4	ชนิดของใบสำคัญส่งของ
CODE 1	C	4	รหัสสินค้า
NAME1	C	10	ชื่อย่อของสินค้า 10 ตัวอักษรแรก
ISSUI	N	6	จำนวนสินค้าที่จ่ายออก
RECI	N	6	จำนวนรับสินค้าเข้า

เมื่อมีการซื้อขายสินค้าใบสำคัญการซื้อขายจะได้รับการนำมากรอกเข้าในไฟล์ DAILY ดังนั้นจึงจำเป็นต้องสร้างไฟล์ให้ง่ายต่อการกรอกข้อความ ในที่นี้ใช้โครงสร้างไฟล์ดังที่กล่าว แต่ที่น่าสังเกตคือ ฟิลด์ CODE1 และ NAM1 นี้มีโครงสร้างเหมือนกับไฟล์มาสเตอร์คือ CODE และ NAME แต่ในไฟล์ DAILY ในส่วนฟิลด์ NAM1 มีเพียง 10 ตัวอักษรแรกเท่านั้น เพราะจะทำให้ง่ายต่อการคีย์ข้อมูลที่ไม่ต้องคีย์ชื่อเต็ม

ในตอนก่อนปิดร้านทุกวันเจ้าของร้านจะต้องทำการปรับปรุงยอดของสินค้าคงคลัง เพื่อให้ใหม่เสมอแต่เพื่อความแน่นอนจึงต้องทำการตรวจสอบว่าไฟล์ DAILY มีการคีย์ข้อมูลผิดพลาดบ้างหรือไม่ วิธีการจะใช้การตรวจสอบว่ามีเรคอร์ดใดบ้างที่ CODE1 กับ NAM1 ไม่ตรงกับ CODE และ NAME ในมาสเตอร์ไฟล์ ในที่นี้ใช้โปรแกรมที่เขียนขึ้นดังโปรแกรมต่อไปนี้

```
B>type check1.cmd
erase
clear
set print on
@ 10,10 say "REXAMINE THESE RECORDS.. CODE AND NAME MISMAATCH"
set print off
use b:point
store poi to pointer
store 1 to count
store "y" to cond
use b:master index cody
select secondary
use b:daily
goto pointer
do while cond<>"n"
  do while count<100
     if pointer>eof
        use b:point
        replace poi with pointer
        return
```

```
        endif
        store code1 to check
        store name1 to rename
        select primary
        find check
        if rename<>$(name,1,10)
           select secondary
           set print on
           display
           set print off
        endif
        store pointer+1 to pointer
        store count+1 to count
        skip
     enddo
     erase
     input "TO CHECK ANOTHER RECORDS Y/N" to cond
     if cond="n".or.cond="no".or.cond="NO".or.cond="N"
        use b:point
        replace poi with pointer
        return
     endif
     store 1 to count
enddo
return
```

จากโครงสร้างของโปรแกรมที่แสดง จะเห็นว่าจะต้องมีการสร้างไฟล์อินเด็กซ์โดยใช้คีย์ CODE ของไฟล์มาสเตอร์เป็นคีย์ฟิลด์ เมื่อเริ่มต้นก็เคลียร์จอภาพและพิมพ์ข้อความ “โปรดตรวจสอบรหัสและชื่อที่ไม่แมทช์” เมื่อใช้อินเด็กซ์ไฟล์โดยให้อินเด็กซ์ไฟล์มาสเตอร์ (คีย์คือ CODE) เป็นไฟล์ปฐมภูมิ และไฟล์ DAILY เป็นไฟล์ทุติยภูมิเมื่อตรวจสอบฟิลด์ CODE กับ CODE1 ที่มีเลขเดียวกันก็จะหาว่าข้อมูลใน NAME กับ NAME1 ตรงกันหรือไม่ ถ้าไม่ตรงก็พิมพ์เรคอร์ดนั้นให้ปรากฏบนจอเพื่อจะได้ตรวจสอบและแก้ไขให้ถูกต้องเสียก่อน โดยใช้คำสั่ง EDIT ในภายหลัง

การสร้างไฟล์ POINT เป็นการเก็บข้อมูลตัวแปรชื่อ POI ไว้เพื่อบอกว่าได้เคยทำการตรวจสอบไฟล์ DAILY ไปเป็นจำนวน POI เรคอร์ดแล้ว ดังนั้นเมื่อทำการตรวจสอบจะได้ทำการตรวจสอบในเรคอร์ดถัดไป (ในกรณีที่ไฟล์ DAILY มีขนาดใหญ่มากอาจทำการตรวจสอบบางส่วนก่อน) การตรวจสอบดังกล่าวจะกระทำเพียงครั้งละ 100 เรคอร์ด สาเหตุที่ทำเช่นนี้ก็เพราะว่าในระหว่างวันอาจมีเวลาว่างก็ทำการตรวจสอบบางส่วนไปก่อนได้ ครั้นถึงเวลาตอนเย็นก่อนปิดร้านก็ตรวจสอบและทำการ UPDATE หมดไฟล์ ดูโปรแกรมการตรวจสอบครั้งเดียวหมดไฟล์ DAILY

โปรแกรมต่อไปนี้เป็นการตรวจสอบไฟล์ครั้งสุดท้ายจนหมดไฟล์

```
B>type check.cmd
erase
clear
set print on
```

```
@ 10,5 say "REXAMINE THESE RECORDS ... CODE AND NAME MISMATCH"
set print off
use b:point
store poi to pointer
use b:master index on code
select secondary
use b:daily
goto pointer
do while .not.eof
  store code1 to check
  store name1 to rename
  select primary
  find for code=check
  if rename<>$(name,1,10)
     select secondary
     set print on
     display
     set print off
     skip
     loop
  endif
  select secondary
enddo
return
```

การทำการตัดสต๊อกหรือ update ไฟล์ใหม่นี้ก็ทำเป็นสองตอนคือ ทำการ update ในระหว่างวันบางส่วนหรือ update ตอนเย็นต่อจนหมดไฟล์ โดยได้แยกโปรแกรมการ update ออกเป็น 2 โปรแกรม

โปรแกรมต่อไปนี้เป็นการทำการตัดสต๊อกและคำนวณยอดการขายในระหว่างวัน

```
erase
clear
use b:point
store pop to pointer
store 1 to count
store "y" to cond
use b:master index cody
select secondary
use b:daily
goto pointer
do while cond<>"n"

   do while count<100
      if pointer>eof
         use b:point
         replace pop with pointer
         return
      endif
```

```
            store code1 to recode
            store reci-issue to add
            select primary
            find recode
            replace onhand with onhand+add
            replace sale with (sale*date+issue)/(date+1)
            replace date with date+1
            if onhand<min
               replace status with "*"
            else
               replace status with " "
            endif
            select secondary
            store "" to recode
            skip
            store pointer+1 to pointer
            store count+1 to count
         enddo
         erase
         input "TO CONTINUE UPDATE MASTER ... Y/N" to cond
         if cond="n".or.cond="no".or.cond="NO".or.cond="N"
            use b:point
            replace pop with pointer
            return
         endif
         store 1 to count
      enddo
      return
```

โปรแกรมต่อไปนี้เป็นการตัดสต๊อกครั้งสุดท้ายกระทำจนหมดไฟล์

```
erase
clear
@ 10,10 say "UPDATE MASTER FILE"
use b:point;store pop to pointer
use b:master index cody
select secondary
use b:daily
goto pop
do while .not.eof

   store code1 to recode
   store issue to iss
   store reci-issue to add
   select primary
```

```
    find recode
    replace sale with (sale*date+iss)/(date+1)
    replace onhand with onhand+add
    if onhand<min
       replace status with "*"
    else
       replace status with " "
    endif
    replace date with date+1
    select secondary
    store "" to recode
    store 0 to iss,add
    skip
enddo
return
```

ในการตัดสต๊อกนั้นจะทำการคำนวณอัตราการขายด้วยโดยใช้สูตรเป็นดังนี้คือ

$$\text{อัตราการขายเมื่อคิดถึงวันที่คำนวณ} = \frac{\text{อัตราการขายวันก่อน} * \text{จำนวนวัน} + \text{ยอดขายวันนี้}}{\text{จำนวนวันเดิม} + 1}$$

นอกจากนี้หากยอดสินค้าเหลือต่ำกว่าที่กำหนดไว้ก็ให้ทำการแสดงสถานะด้วยเครื่องหมาย * เพื่อบอกให้ทราบว่าควรสั่งสินค้าชนิดนี้ข้าสต๊อกเพิ่มได้อีกแล้วเป็นการเตือนผู้มีหน้าที่เกี่ยวข้องได้ทันท่วงที

ที่กล่าวมานี้เป็นวิธีการกระทำอย่างไม่ยุ่งยากนัก แต่ก็มีข้อจำกัดหลายประการ เช่น ไม่สามารถทราบลักษณะการไหลของสินค้าว่ามีการเข้าออกถี่ห่างเพียงไร

ตัวอย่างการประยุกต์ทำบัญชีสมุดสั่งจ่ายเช็ค

ตัวอย่างนี้เป็นเพียงตัวอย่างการเขียนโปรแกรมโครงสร้างสำหรับดีเบสทูเพื่อใช้งานในระบบจัดการสมุดเช็ค โดยมีโครงสร้างการทำงานที่เข้าใจได้ง่ายเพื่อให้เห็นตัวอย่างการใช้งานโดยมีการใช้ไฟล์เพียงไฟล์เดียวคือ ไฟล์ CHECKREG ซึ่งเก็บข้อมูลรายละเอียดของสมุดเช็คโดยมีการสร้างไฟล์ดังโปรแกรมต่อไปนี้

```
. CREATE
ENTER FILENAME: CHECKREG
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD    NAME,TYPE,WIDTH,DECIMAL PLACES
 001      NO,N,4
 002      TO,C,30
 003      AMT,N,10,2
 004      CAN,L
 005      DATE,C,10
```

```
 006
INPUT DATA NOW? N
```

โครงสร้างของไฟล์พออธิบายอย่างคร่าว ๆ ได้ดังนี้

NO หมายถึง หมายเลขเช็คที่ได้ทำการสั่งจ่ายเป็นเลข 4 หลัก

TO หมายถึง ผู้รับเช็คตามที่เขียนลงใบเช็คใบนี้เป็นตัวอักษร 30 ตัว

AMT หมายถึง จำนวนเงินที่จ่ายในเช็คใบนั้นเป็นตัวเลข 10 หลักทศนิยม 2 ตำแหน่ง

CAN หมายถึง สถานะของเช็คใบนั้นว่ามีการยกเลิกหรือไม่

DATE หมายถึง วันที่ที่ลงในเช็คใบนั้นเป็นตัวอักษร 10 ตัว

ในการใช้งานสำหรับไฟล์นี้จะเริ่มด้วยสิ่งที่ต้องการ ในที่นี้คือเมื่อมีการสั่งจ่ายเช็คก็จะมีการป้อนข้อมูลเพื่อเพิ่มเรคอร์ดในไฟล์ และถ้าหากยกเลิกเช็คก็จะต้องบอกสถานะให้ทราบ และเมื่อมีการฝากเงินเข้าบัญชีเพิ่มก็จะทำการปรับยอดเงินคงเหลือในบัญชีเช็คนั้น ในที่นี้จึงใช้โหมดการทำงานตามเมนูดังโปรแกรมต่อไปนี้

```
*
SET TALK OFF
USE CHECKREG
DO WHILE T
   ?
   ?
   ?
   ? '            Checkbook Balancer Menu'
   ?
   ? '            0 - EXIT'
   ? '            1 - Enter New Checks'
   ? '            2 - Enter Cancelled Checks'
   ? '            3 - Balance'
   ?
   ? ' enter desired action'
   WAIT TO ACTION
   IF ACTION ='0'
      SET TALK on
      CANCEL
   ENDIF
   IF ACTION ='1'
      DO NEWENTR
   ENDIF
   IF ACTION ='2'
      DO CANCELS
   ENDIF
   IF ACTION ='3'
```

```
            DO BALANCE
        ENDIF
    ENDDO
    RETURN
```

โปรแกรมเมนูนี้เริ่มจากการใช้ไฟล์ CHECKREG แล้วพิมพ์ข้อความเมนูด้วยคำสั่ง ? หรือพิมพ์ โดยคำสั่งนี้ไม่สามารถกำหนดตำแหน่งแถวและคอลัมน์ได้ คือเป็นการพิมพ์เรียงกันไป คำสั่ง WAIT TO ACTION หมายถึงรอค่าการกดคีย์ และถ้ามีการกดคีย์ก็จะรับเข้ามาในตัวแปรชื่อ ACTION ถ้า ACTION = 0 ก็จะทำคำสั่ง CANCEL คือเลิกทำโปรแกรมนี้และออกจากโปรแกรม ถ้าคีย์ 1 ก็จะทำตามโปรแกรม NEWENTR คือการป้อนข้อมูลเช็คใหม่ ถ้าคีย์ 2 ก็จะทำโปรแกรมชื่อ CANCELS คือการป้อนข้อมูลยกเลิกเช็ค และถ้าคีย์ 3 ก็จะทำตามโปรแกรม BALANCE

พิจารณาจากโปรแกรมทั้งสามคือ

โปรแกรมต่อไปนี้เป็นโปรแกรม NEWENTR. CMD ซึ่งเป็นการป้อนข้อมูลเช็คที่มีการสั่งจ่ายใหม่

```
*
REMARK Enter Check Number of 0 to Exit
DO WHILE T
   ?
   ?
   INPUT "Enter Check Number           " to C:NO
   IF C:NO=0
      RETURN
   ENDIF
   ?
   ACCEPT "Paid to Order of        " to C:TO
   INPUT "Amount of Check          " to C:AMT
   ACCEPT "DATE of Check           " to C:DAT
   ?
   INPUT "Are all fields correct ? " to GO:NOGO
   IF .NOT.GO:NOGO
      LOOP
   ENDIF
   APPEND BLANK
   REPLACE NO with C:NO, TO with C:TO, AMT with C:AMT, DATE ;
     with C:DAT, CAN with F
ENDDO
```

โปรแกรมนี้เริ่มจากการกระทำโดยการป้อนหมายเลขเช็ค ถ้าหมายเลขเช็ค = 0 จะหมายถึงเสร็จสิ้นจากโปรแกรมนี้และกลับไปโปรแกรมเมนูใหม่ เมื่อป้อนหมายเลข 0: เก็บไว้ในตัวแปรก่อนคือ C:NO, C:TO, C:AMT และ C:DAT เมื่อป้อนเสร็จแล้วก็จะถามว่ากรอกข้อมูลทุกฟิลด์ถูกต้องหรือไม่ โดยใช้ตัวแปร GO: NO GO ถ้าตัวแปรนี้เป็นลอจิกเท็จ (N,F) ก็จะกลับไปกระทำ "Enter Check number"

ใหม่ โดยไม่มีการเก็บข้อมูลในไฟล์ แต่ถ้าเป็น Y หรือ T ก็จะสร้างเรคอร์ดใหม่ แล้วกลับไปรับคีย์เริ่มการป้อนข้อมูลเรคอร์ดต่อไป รอจนกว่าจะมีการคีย์ C : NO ด้วย "0" จึงกลับโปรแกรมเมนู

พิจารณาจากโปรแกรม CANCELS. CMD

โปรแกรมต่อไปนี้เป็นโปรแกรม CANCELS จะเป็นการป้อนสถานะว่าเช็คใบใดยกเลิก

```
*
REMARK ENTER Check Number of 0 to Exit
DO WHILE T
    ?
    INPUT "Enter Cancelled Check no " to C:CAN
    IF C:CAN=0
       RETURN
    ENDIF
    GO TOP
    LOCATE for C:CAN=NO
    REPLACE CAN WITH t
ENDDO
```

โปรแกรมนี้ก็รับคีย์โดยคีย์หมายเลขเช็คใบที่ต้องการยกเลิก แต่ถ้าคีย์หมายเลข "0" จะกลับเข้าเมนูใหม่ แต่ถ้าไม่เป็น "0" ก็จะทำการใส่สถานะลอจิกเป็นใบฟิลด์ CAN ด้วยลอจิก T

โปรแกรมสุดท้ายของตัวอย่างนี้คือโปรแกรม BALANCE. CMD โปรแกรมนี้จะอ่านข้อมูลจำนวนเงินที่ฝากเข้าบัญชีแล้วทำการคำนวณโดยรวมยอดเงินฝากกับเงินจ่ายตามเช็คใบที่ไม่ยกเลิก

```
*
SUM AMT to OUTSTAND for .NOT.CAN
?
?
DISP off  'Total Outstanding Checks = $',OUTSTAND
?
REMARK Enter Outstanding Deposits, Enter 0 to Proceed
STORE T TO ACTIVE
STORE 1 TO COUNT
STORE 0 TO T:OUT
DO WHILE ACTIVE
   STORE STR(COUNT,3) to I
   INPUT 'Enter Amount of Outstanding Deposit &I ' to D:OUT
   IF D:OUT=0
      STORE F to ACTIVE
   ELSE
      STORE D:OUT+T:OUT to T:OUT
      STORE COUNT+1 to COUNT
   ENDIF
ENNDO
DISPLAY OFF COUNT-1,' Total Outstanding Deposits   Total  = $' ,T:OUT
?
INPUT "Enter Ending Balance" to BEGIN
DISPLAY OFF 'Current Balance = $',BEGIN+T:OUT-OUTSTAND
WAIT
RETURN
```

โปรแกรมเริ่มจากการรวมจำนวนเงินในใบเช็คที่ไม่ได้ยกเลิกไว้ในตัวแปร OUTSTAND จากนั้นแสดงผลจำนวนเงิน หลังจากนั้นจะเข้าลูปของการฝากเงินโดยบอกว่าฝากครั้งที่ 1,2,3 ถ้าเงินฝากเป็น 0 จะออกจากลูปไปคำนวณเงิน เงินฝากจะป้อนเข้าตัวแปร D:OUT และบวกทบใน T : OUT เมื่อเสร็จสิ้นการป้อนเงินฝากก็จะนำมาคำนวณโดยถามว่าเงินฝากเดิมคือ BEGIN เป็นเท่าไรที่ให้เราบอก จากนั้นจะมีการคำนวณแล้วแสดงผล เห็นไหมครับตัวอย่างนี้มีหลักการไม่ยากเลย ลองพิจารณาโปรแกรมและการทำงาน-ดังต่อไปนี้

```
.
. DO MENU


          Checkbook Balancer Menu

          0 - EXIT
          1 - Enter New Checks
          2 - Enter Cancelled Checks
          3 - Balance

 enter desired action
WAITING 2
 ENTER Check Number of 0 to Exit


Enter Cancelled Check no :
DO CANCELLED
.
.
.
.
.
.
.
.
.
.
.
.
.
. DO MENU


          Checkbook Balancer Menu
```

```
          0 - EXIT
          1 - Enter New Checks
          2 - Enter Cancelled Checks
          3 - Balance

 enter desired action
WAITING 1
 Enter Check Number of 0 to Exit


Enter Check Number                :1000

Paid to Order of           :TANAKUL CO.
Amount of Check             :123.45
DATE of Check               :10 JUN 83

Are all fields correct ? :Y


Enter Check Number                :1001

Paid to Order of           :HENG HUAD PUBLISHING CO.
Amount of Check             :79.88
DATE of Check               :12 JUN 83

Are all fields correct ? :Y


Enter Check Number                :1003

Paid to Order of           :SILOM COMM. INC.
Amount of Check             :723.31
DATE of Check               :14 JUN 83

Are all fields correct ? :Y


Enter Check Number                :1004

Paid to Order of           :THAI SANGYONT
Amount of Check             :2753.77
DATE of Check               :19 JUN 79

Are all fields correct ? :Y


Enter Check Number                :0
```

```
          Checkbook Balancer Menu

          0 - EXIT
          1 - Enter New Checks
          2 - Enter Cancelled Checks
          3 - Balance

 enter desired action
WAITING 2
 ENTER Check Number of 0 to Exit

Enter Cancelled Check no :1001

Enter Cancelled Check no :1003

Enter Cancelled Check no :0


          Checkbook Balancer Menu

          0 - EXIT
          1 - Enter New Checks
          2 - Enter Cancelled Checks
          3 - Balance

 enter desired action
WAITING 3


Total Outstanding Checks = $     2877.22

 Enter Outstanding Deposits, Enter 0 to Proceed
Enter Amount of Outstanding Deposit   1 :1234.56
Enter Amount of Outstanding Deposit   2 :3.25
Enter Amount of Outstanding Deposit   3 :333.50
Enter Amount of Outstanding Deposit   4 :0
    3  Total Outstanding Deposits Total = $ 1571.31

Enter Ending Balance:1445.89
Current Balance = $        139.98
WAITING



       Checkbook Balancer Menu

       0 - EXIT
```

```
1 - Enter New Checks
2 - Enter Cancelled Checks
3 - Balance

enter desired action
WAITING 0
DO CANCELLED
```

เมื่อร้านขายยาอยากใช้ไมโครคอมพิวเตอร์

ตัวอย่างนี้เป็นตัวอย่างที่ออกแบบมาเพื่อมีจุดมุ่งหมายให้เห็นวิธีการใช้ดีเบสทูในการสร้างไฟล์แบบง่าย ๆ การป้อนข้อมูล การแก้ไข การค้นหาข้อมูล และทำรายงาน โดยสมมติว่าร้านขายยาแห่งหนึ่งต้องการใช้ไมโครคอมพิวเตอร์เก็บรวบรวมรายชื่อยาและจำนวนยาที่เหลืออยู่ แต่เนื่องจากชื่อยามีชื่อเรียกหลายอย่างจึงต้องสร้างไฟล์ให้ง่ายต่อการค้นหาโดยมีโครงสร้างไฟล์ดังนี้

ชื่อสามัญ	CONAME	30, C
ชื่อการค้า	GRNAME	30, C
ชื่อการนำเข้า	IMNAME	30, C
ชื่อยาต้นตำหรับ	SONAME	30, C
จำนวนยา	QUANTITY	6, N
หน่วยจำนวน	UNIT	8, C
มูลค่า	VALUE	N, 10, 2
ราคา	PRICE	N, 10, 2

เมื่อสร้างไฟล์ได้แล้วสิ่งที่ต้องการของเจ้าของร้านขายยาคือ ต้องการป้อนข้อมูล ตรวจสอบแก้ไขข้อมูล ค้นหาและเลือกทำรายงาน ในที่นี้จึงทำเป็นเมนูให้เจ้าของร้านดังโปรแกรมต่อไปนี้

```
A>type main.cmd
set talk off
use b:drug
store t to c
do while c
   erase
   ? '    ****************************************'
   ? '    *                                      *'
   ? '    *              MENU                    *'
   ? '    *                                      *'
   ? '    *      1 - ENTER   INFORMATION         *'
   ? '    *                                      *'
   ? '    *      2 - EDIT    INFORMATION         *'
   ? '    *                                      *'
   ? '    *      3 - SELECT  INFORMATION         *'
   ? '    *                                      *'
   ? '    *      4 - end                         *'
   ? '    *                                      *'
   ? '    ****************************************'
   ?
   ? '        *PLEEASE ENTER YOUR SELECTION*'
   ?
```

```
   wait to choice
   do case
      case choice='1'
           do b:enter
      case choice='2'
           do b:ed
      case choice='3'
           do b:sele
      case choice='4'
           quit
   endcase
enddo
```

จากโปรแกรมเมนูนี้แบ่งออกเป็น 3 โปรแกรมย่อยกันอีก 1 ที่เป็นคำสั่ง QUIT คือ เลิก โปรแกรมย่อยที่เรียกคือ enter, ed และ sele

ลองมาพิจารณาโปรแกรมย่อยที่ใช้ในการป้อนข้อมูล โดยจุดมุ่งหมายคือต้องการให้เจ้าของร้านที่ไม่รู้เรื่องเกี่ยวกับคอมพิวเตอร์ได้เข้าใจในเรื่องการป้อนข้อมูลหรือวิธีใช้ได้ง่ายจึงทำการป้อนข้อมูลในรูปแบบที่เป็นการแสดงตามความหมายในลักษณะ full screen format ดังกล่าว เมื่อลองพิจารณาดูโปรแกรมต่อไปนี้

```
A>type enter.cmd
do while c
   store ' ' to a
   append blank
   erase
   @ 2,10 say 'ENTER INFORMATION'
   @ 4,5 say 'MONTH:'
   @ 6,5 say 'YEAR:'
   @ 8,5 say 'COMMON-NAME:'
   @ 10,5 say 'TRADE-NAME:'
   @ 12,5 say 'IMPORTER-NAME:'
   @ 14,5 say 'SOURCE-NAME:'
   @ 16,5 say 'QUANTITY:'
   @ 18,5 say 'UNIT:'
   @ 20,5 say 'VALUE:'
   @ 4,10 get month
   read
   if month=0
      delete
      pack
      return
   endif
   @ 6,9 get year
   @ 8,16 get coname
   @ 10,15 get trname
   @ 12,18 get imname
   @ 14,16 get soname
   @ 16,13 get quantity
```

```
    @ 18,9 get unit
    @ 20,10 get value
    read
    @ 22,5 say 'CORRECT y/n' get a
    read
    if !(a)='N'
       delete
       loop
    endif
    replace price with value/quantity
enddo
```

เมื่อเริ่มโปรแกรมก็ใช้การกำหนดตัวแปร a เป็น " " แล้วเปิดเรคอร์ดว่างที่จะต่อท้าย จากนั้นพิมพ์รูปแบบข้อความต่าง ๆ บนจอในตำแหน่งที่สวยงามตามต้องการอาจพิมพ์ข้อความ 5 หรือรายละเอียดเพิ่มเติมอย่างไรก็ได้ ส่วนข้อความ 4, 10 get month หมายถึงตำแหน่งเคอร์เซอร์เมื่อแสดงข้อความหมดแล้วจะมาหยุดอยู่ที่ตำแหน่ง 4, 10 ซึ่งเป็นหลังข้อความว่า MONTH เพราะ MONTH เริ่มพิมพ์บรรทัดที่ 4 คอลัมน์ที่ 5 เมื่อมีการคีย์ข้อมูลเข้าก็จะเก็บที่ MONTH ถ้าเป็น 0 ก็จะลบเรคอร์ดสุดท้ายที่เปิดมานี้เสีย แล้วกลับโปรแกรมเมนู หาก MONTH ไม่เป็น 0 เคอร์เซอร์จะมาในตำแห่ง 6, 9 เพื่อรับคีย์ และเมื่อ RETURN เคอร์เซอร์ตกลงมาที่ 8, 16 เพื่อพิมพ์ชื่อสามัญ จะเห็นว่าตำแหน่งการพิมพ์จะตรงกับข้อความที่แนะนำไว้ การคีย์ข้อความแต่ละเรคอร์ดถ้าผิดพลาดก็เคลื่อนเคอร์เซอร์ไปแก้ไขได้ เมื่อพ้นข้อความ read ก็จะได้รับการอ่านเข้าไปแต่เครื่องจะถามเพื่อแน่ใจอีกครั้ง ถ้าหากไม่ต้องการก็ลบเรคอร์ดนี้ทิ้งได้

สำหรับไฟล์ที่ป้อนข้อมูลไปแล้วเราก็สามารถทำการแก้ไขได้โดยเจ้าของร้านไม่ต้องรู้อะไรเลย เพียงเมื่อขึ้น MENU ก็เลือกหมายเลข 2 ซึ่งจะมากระทำตามโปรแกรม ed. cmd นี้ ลองพิจารณาโปรแกรมต่อไปนี้

```
A>type ed.cmd
erase
store '
store 0 to mo
store 0 to ye
@ 2,10 say 'EDIT INFORMATION'
@ 4,5 say 'MONTH' get mo picture '99'
@ 5,5 say 'YEAR' get ye picture '9999'
@ 6,5 say 'COMMON-NAME' get name
read
go top
locate all for (month=mo.and.year=ye.and.coname=name)
do while .not. eof
```

```
    store ' ' to code
    @ 7,5 say 'CODE' get code
    @ 7,16 say '(1=CORRECT 2=DELETE)'
    @ 10,5 say 'TRADE-NAME:'+trname+':'
    @ 12,5 say 'IMPORTER-NAME:'+imname+':'
    @ 14,5 say 'SOURCE-NAME:'+soname+':'
    @ 16,5 say 'COMMON-NAME:'+coname+':'
    @ 18,5 say 'QUANTITY:'+str(quantity,6)+':'
    @ 18,14 say quantity using '999,999'
    @ 20,5 say 'UNIT:'+unit+':'
    @ 22,5 say 'VALUE:'
    @ 22,11 say value using '9,999,999,999.99'
    read
    if code='2'
       delete
       continue
       loop
    endif
    @ 18,14 say '          '
    @ 22,11 say '                '
    @ 10,15 get trname
    @ 12,18 get imname
    @ 14,16 get soname
    @ 16,16 get coname
    @ 18,13 get quantity
    @ 20,9 get unit
    @ 22,10 get value
    read
    replace price with value/quantity
    continue
enddo
pack
return
```

โปรแกรมการแก้ไขมีจุดมุ่งหมายให้เจ้าของร้านขายยาแก้ไขได้โดยไม่ต้องรู้อะไรในดีเบสเลย ในการแก้ไข เครื่องจะพิมพ์ข้อความออกมาเพื่อถามว่าถึงเดือนปีและชื่อสามัญ โดยเดือนและปีมีโครงสร้างต้องเป็นการแก้ไข เครื่องจะพิมพ์ข้อความออกมาเพื่อถามว่าถึงเดือนปีและชื่อสามัญ โดยเดือนและปีมีโครงสร้างต้องเป็นตัวเลข เมื่อบอกข้อความนี้เครื่องจะค้นหาเรคอร์ดที่มีรายละเอียดดังกล่าวพร้อมกับพิมพ์ข้อความในเรคอร์ดนั้นทั้งหมดออกมาตามต้องการ และเครื่องจะถามว่าจะแก้หรือลบทิ้ง โดยผู้ใช้ต้องตอบ 1 คือแก้ และ 2 คือลบทิ้งถ้าคีย์ในทั้งสองแบบนี้เครื่องคอมพิวเตอร์จะกระทำตามโดยการแก้นั้นสามารถเลื่อนเคอร์เซอร์ขึ้นมาแก้ไขข้อความที่พิมพ์ไว้ได้เลย โดยใช้คำสั่งคีย์ตามโหมด EDIT ทุกประการ

โปรแกรมสุดท้ายคือการค้นหาข้อมูล การค้นหาสามารถค้นหาได้ตามชื่อทั้งชื่อทางการค้า ชื่อสามัญ ชื่อการสั่งเข้า และชื่อต้นตำหรับ ลองดูโปรแกรมต่อไปนี้

```
erase
store '                              ' to name
store 0 to mo
store 0 to ye
store ' ' to se
store 'N' to p
@ 4,10 say 'ENTER SELECTION'
@ 6,5 say 'MONTH' get mo picture '99'
@ 8,5 say 'YEAR' get ye picture '9999'
@ 12,5 say 'YOUR SELECTION' get se
@ 14,3 say '(1= COMMON-NAME  2=TRADE-NAME)'
@ 15,3 say '(3=IMPORTER-NAME 4=SOURCE-NAME)'
@ 18,5 say 'NAME' get name
read
do case
   case se='1'
        store 'coname' to na
   case se='2'
        store 'trname' to na
   case se='3'
        store 'imname' to na
   case se='4'
        store 'soname' to na
   otherwise
        return
endcase
store 20 to line
store 1 to page
store 'erase' to com
if mo=0.or.ye=0
   do b:head1
   do b:rep1
else
   do b:head2
   do b:rep2
endif
? '*********************'
? '*  TO PRINTER Y/N ? *'
? '*********************'
wait to p
if !(p)='Y'
   set format to print
   store 60 to line
   store 1 to page
   store 'eject' to com
   if mo=0.or.ye=0
      do b:head1
      do b:rep1
```

```
   else
      do b:head2
      do b:rep2
   endif
   set format to screen
   store 'N' to p
endif
return
```

เมื่อเจ้าของร้านพิมพ์ทางเลือก 1, 2, 3, 4 และวันที่แล้ว เครื่องก็จะทำรายงานให้โดยเป็นรายงานที่พิมพ์มีหัวกระดาษมีเลขหน้า โดยการพิมพ์จะพิมพ์ 2 รูปแบบคือ ถ้าวันที่เดือนเป็น 0, 0 จะพิมพ์หัวกระดาษตาม head1 และรายงานตาม rep1 แต่ถ้าเป็นอย่างอื่นจะพิมพ์หัวกระดาษเป็น head2 และรายงานเป็น rep2 ขอให้ลองศึกษารูปแบบการพิมพ์หัวกระดาษรายงานและตัวรายงาน การกำหนดบรรทัดต่อหน้า เป็นต้น

โปรแกรม head 1

```
&com
@ 1,60 say '*** LIST DETAIL  ***'
@ 1,110 say 'page no. '+str(page,2)
@ 4,2 say ' MONTH'
@ 4,10 say 'COMMON NAME    '
@ 4,27 say 'TRADE NAME     '
@ 4,44 say 'IMPORTER NAME '
@ 4,61 say 'SOURCE NAME    '
@ 4,77 say 'QUANTITY'
@ 4,87 say 'UNIT'
@ 4,97 say ' UNITPRICE'
@ 4,109 say '           VALUE'
@ 5,2 say ' /YEAR'
return
```

โปรแกรม rep 1

```
store 0 to qtotal
store 0 to vtotal
store 6 to row
go top
locate all for &na=name
do while .not.eof
   if row>line
      store 6 to row
      store page+1 to page
      if !(p)<>'Y'
         wait
      endif
```

```
            do b:head1
        endif
        @ row,2 say str(month,2)+'/'+str(year,4)
        @ row,10 say $(coname,1,15)
        @ row,27 say $(trname,1,15)
        @ row,44 say $(imname,1,15)
        @ row,61 say $(soname,1,15)
        @ row,78 say quantity using '999,999'
        @ row,87 say unit
        @ row,97 say price using '999,999.99'
        @ row,109 say value using '9,999,999,999.99'
        @ row+1,10 say $(coname,16,15)
        @ row+1,27 say $(trname,16,15)
        @ row+1,44 say $(imname,16,15)
        @ row+1,61 say $(soname,16,15)
        store row+2 to row
        store qtotal+quantity to qtotal
        store vtotal+value to vtotal
        continue
    enddo
    @ row,59 say 'TOTAL QUANTITY:'
    @ row,74 say qtotal using '999,999,999'
    @ row,96 say 'TOTAL VALUE:'
    if vtotal>99999999
        @ row,108 say val(str(vtotal,15,2)):
    using '99,999,999,999.99'
    else
        @ row,108 say vtotal using '99,999,999,999.99
    endif
    return
```

โปรแกรม head 2

```
&com
@ 1,60 say '*** LIST DETAIL  ***'
@ 1,110 say 'page no. '+str(page,2)
@ 2,57 say '** MONTH '+str(mo,2)+' ,YEAR '+str(ye,4)+' **'
@ 4,10 say 'COMMON NAME    '
@ 4,27 say 'TRADE NAME     '
@ 4,44 say 'IMPORTER NAME '
@ 4,61 say 'SOURCE NAME    '
@ 4,77 say 'QUANTITY'
@ 4,87 say 'UNIT'
@ 4,97 say ' UNITPRICE'
@ 4,109 say '              VALUE'

return
```

โปรแกรม rep 2

```
store 0 to qtotal
store 0 to vtotal
store 6 to row
go top
locate all for (month=mo.and.year=ye.and.&na=name)
do while .not.eof
   if row>line
      store 6 to row
      store page+1 to page
      if !(p)<>'y'
         wait
      endif
      do b:head2
   endif
   @ row,10 say $(coname,1,15)
   @ row,27 say $(trname,1,15)
   @ row,44 say $(imname,1,15)
   @ row,61 say $(soname,1,15)
   @ row,78 say quantity using '999,999'
   @ row,87 say unit
   @ row,97 say price using '999,999.99'
   @ row,109 say value using '9,999,999,999.99'
   @ row+1,10 say $(coname,16,15)
   @ row+1,27 say $(trname,16,15)
   @ row+1,44 say $(imname,16,15)
   @ row+1,61 say $(soname,16,15)
   store row+2 to row
   store qtotal+quantity to qtotal
   store vtotal+value to vtotal
   continue
enddo
@ row,59 say 'TOTAL QUANTITY:'
@ row,74 say qtotal using '999,999,999'
@ row,96 say 'TOTAL VALUE:'
if vtotal>99999999
   @ row,108 say val(str(vtotal,15,2)) using '99,999,999,999.99'
else
   @ row,108 say vtotal using '99,999,999,999.99'
endif
return
```

ตัวอย่างโปรแกรมควบคุมสินค้าคงคลัง

ตัวอย่างนี้ใช้โครงสร้างไฟล์หลักเหมือนตัวอย่างแรกดังได้กล่าวไว้แล้ว กล่าวคือ ในโปรแกรมนี้ใช้ไฟล์ทั้งหมดคือ ไฟล์ MASTER ไฟล์อินเด็กซ์ในฟิลด์ CODE ของไฟล์มาสเตอร์ ไฟล์ POINT เก็บค่าพอยท์เตอร์และไฟล์ DAILY

ลักษณะการนำเข้าสู่ระบบมีทางเลือก 9 ทาง โดยทางเลือกแต่ละทางจะมีจุดมุ่งหมายแตกต่างกันตามความต้องการของผู้ใช้ ทางเลือกต่าง ๆ เป็นดังนี้

ทางเลือก 0 คือการ list ไฟล์ฐานข้อมูลออกมาดู

ทางเลือก 1 เป็นการป้อนข้อมูลเข้าไฟล์ daily

ทางเลือก 2 เป็นการตรวจสอบข้อมูล

ทางเลือก 3 ทำการ update กับไฟล์มาสเตอร์

ทางเลือก 4 พิมพ์รายงาน

ทางเลือก 5 ทำการแก้ไขข้อมูลในฐานข้อมูล

ทางเลือก 6 เลิกโปรแกรม

ทางเลือก 7 ตระเตรียมงานในวันรุ่งขึ้น

ทางเลือก 8 กลับสู่ดีเบสทู

เมื่อมีการนำข้อมูลเข้าสู่ไฟล์ daily แล้ว ไม่ว่าในกรณีที่นำข้อมูลเข้าหมดแล้ว (กรณีเลิกงานในวันนั้น) หรือขณะที่ทำงานอยู่ก็ตามจะมีการเช็คความถูกต้อง อาจจะโดยผู้ทำงานตรวจเอง หรือใช้โปรแกรมตรวจให้ ซึ่งในโปรแกรมที่จัดทำจะตรวจสอบดูว่าชื่อและรหัสตรงกันหรือไม่ และโปรแกรมจะพิมพ์รหัสไว้ในส่วนของเรคอร์ดที่ผิดพลาด การเช็คข้อมูลไม่จำเป็นต้องเช็คครั้งเดียวจบก็ได้ เมื่อข้อมูลได้ตรวจสอบแล้วก็จะนำเข้ามาทำการ update โดยจะแบ่งการทำทีละส่วนหรือทั้งหมดก็ได้ และถ้าต้องการรายงานผลก็จะรายงานผลออกมา โดยปกติจะพิมพ์รายงานผลของไฟล์มาสเตอร์บ่อยเท่าไรก็ได้

การกระทำทั้งหมดจะเป็นไปตามโปรแกรมต่อไปนี้

```
type ku.cmd
*********************************************************************
* INVENTORY CONTROL PROGRAM   ......                                *
*   BY MR.NARONGPON PANTONG   .. STUDENT OF ENGINEERING FACULTY     *
*     KASETSART UNIVERSITY     BANGKOK  THAILAND                    *
*       17 APRIL 1983                                               *
*********************************************************************
erase
clear
set talk off
store "y" to new
* SELECT MODE OF OPERATION
do while new<>"n"
   store "a" to way,soi,kk
   do while way<>"1".and.way<>"2".and.way<>"3".and.way<>"4".and.way<>"5";
        .and.way<>"6".and.way<>"7".and.way<>"8".and.way<>"0"
      erase
      ? chr(7)
      @ 04,5 say "****************************************"
      @ 05,5 say "****************************************"
      @ 06,5 say "*                                      *"
      @ 07,5 say "* SELECT No. OF YOUR COMMAND ...       *"
      @ 08,5 say "*        0 ... TO LIST DATA BASE FILES *"
      @ 09,5 say "*        1 ... TO APPEND DATA          *"
      @ 10,5 say "*        2 ... TO CHECK DATA           *"
      @ 11,5 say "*        3 ... TO UPDATE MASTER        *"
      @ 12,5 say "*        4 ... TO PRINT REPORT         *"
      @ 13,5 say "*        5 ... TO EDIT DATA BASE FILES *"
      @ 14,5 say "*        6 ... TO QUIT                 *"
      @ 15,5 say "*        7 ... TO PREPARE FOR TOMORROW *"
      @ 16,5 say "*        8 ... TO RETURN TO NORMAL DBASE *"
      @ 17,5 say "*                                      *"
      @ 18,5 say "****************************************"
      @ 19,5 say "****************************************"
      ?
      wait to way
      ? chr(7)
```

```
enddo
*LIST DATA BASE FILES
if way="0"
   use
   list
   *DELAY
   store 1 to ss
   do while ss<100
      store ss+1 to ss
   enddo
   store "y" to new
   loop
endif
* APPEND DATA ROUTINE
if way="1"
   erase
   clear
   store "n" to KK
   do while KK<>"3".and.KK<>" "
      @ 05,5 say "......................................."
      @ 06,5 say ".                                     ."
      @ 07,5 say ". SELECT ...                          ."
      @ 08,5 say ".           3 ... TO CANCEL COMMAND   ."
      @ 09,5 say ". SPCCE BAR ... TO CONTINUE           ."
      @ 10,5 say ".                                     ."
      @ 11,5 say "......................................."
      ?
      wait to KK
      ? chr(7)
   enddo
   if KK="3"
      store "y" to new
      loop
   endif
   use b:daily
   store "y" to ar
   do while ar="y"
      append blank
      erase
      ? "             ***   APPEND DATA   ***  "
      @ 5,2 say "PRESS REUTRN TO STOP APPEND"
      @ 7,2 say "CODE                 " get code1
      read
      if code1=chr(32)
         delete
         pack
         store "n" to ar
         loop
      endif
      @ 09,2 say "ITEM.No.          " get item
      @ 11,2 say "TRANSACTION.No " get trans
      @ 13,2 say "NAME              " get name1
      @ 15,2 say "ISSUE             " get issue
      @ 17,2 say "RECIEVE           " get reci
      read
      loop
   enddo
   store "y" to new
   loop
endif
* CHECK DAILY FILE ROUTINE
if way="2"
   store "a" to soi
   do while soi<>"1".and.soi<>"2".and.soi<>"3"
      erase
```

```
        @ 05,5 say"......................................."
        @ 06,5 say".                                     ."
        @ 07,5 say".  SELECT...                          ."
        @ 08,5 say".        1 ... TO USE SECTION CHECK   ."
        @ 09,5 say".        2 ... TO USE FINAL CHECK     ."
        @ 10,5 say".        3 ... TO CANCEL COMMAND      ."
        @ 11,5 say".                                     ."
        @ 12,5 say"......................................."
        ?
        wait to soi
        ? chr(7)
     enddo
     if soi="3"
        store "y" to new
        loop
     endif
     do check
     store "y" to new
     loop
  endif
  * UPDATE MASTER FILE ROUTINE
  if way="3"
     do b:update
     store "y"to new
     loop
  endif
  * PRINT REPORT HERE
  if way="4"
     do report
     store "y" to new
     loop
  endif
  * EDITING DATA BASE FILES
  if way="5"
     store "a" to kk
     do while kk<>"3".and.kk<>" "
        erase
        @ 05,5 say"-------------------------------"
        @ 07,5 say"SELECT ..."
        @ 08,5 say"         3 ... TO CANCEL COMMAND"
        @ 09,5 say"SPACE BAR ... TO CONTINUE"
        @ 11,5 say"-------------------------------"
        ?
        wait to kk
        ? chr(7)
     enddo
     if kk="3"
        store "y" to new
        loop
     endif
     erase
     @ 05,5 say"-----------------------------------------------"
     @ 07,5 say"           ***** EDITING COMMAND *****"
     @ 08,5 say" .. CTRL-R TO GO TO PREVIOUS RECORDS"
     @ 09,5 say" .. CTRL-C TO GO TO NEXT RECORDS"
     @ 10,5 say" .. CTRL-W TO STOP EDITING"
     @ 11,5 say" .. CTRL-U TO TOGGLE BETWEEN DELETE AND RECALL"
     @ 12,5 say"             ***** GOOD LUCK *****"
     @ 14,5 say"-----------------------------------------------"
     ?
     ? chr(7)
     store "k" to dasie
     accept "TELL ME FILES' NAME " to dasie
     input "ENTER YOUR FIRST RECORD TO EDIT" TO NUM
     use &dasie
     go num
```

```
    edit num
    pack
    store "y" to new
    loop
 endif
 if way="6"
    quit
 endif
 * PREPARE FOR THE NEXT DAY.USE THIS SECTION AFTER YOU HAVE FINISHED ALL WORK
 if way="7"
    store "a" to soi
    do while soi<>"3".and.soi<>" "
       erase
       @ 02,1 say"---------------------------------------------------"
       @ 03,1 say"                        CAUTION "
       @ 05,1 say"* AFTER THIS COMMAND WORK DAILY FILE WILL DELETE *"
       @ 07,1 say" ** DO THIS SECTION AFTER FINISH ALL WORK ONLY **"
       @ 07,5 say"   SELECT ..."
       @ 08,5 say"            3 ... TO CANCEL COMMAND"
       @ 09,5 say"   SPACE BAR ... TO CONTINUE"
       @ 11,1 say"---------------------------------------------------"
       ?
       wait to soi
       ? chr(7)
    enddo
    if soi="3"
       store "y" to new
       loop
    endif
    use b:daily
    delete all
    pack
    use b:master
    store date to da
    replace all date with da+1
    pack
    use pointer
    replace poi with 1,pop with 1
    pack
    quit
 endif
 if way="8"
    set talk on
    return
    ? chr(7)
 endif
enddo
*****************************************************
********** END OF INVENTORY CONTROL PROGRAM **********
****** KU.MICROCOMPUTER RESEARCH LAB PRODUCTION ******
*************** KASETSART UNIVERSITY ****************
*****************************************************
```

```
A>type update.cmd
* UPDATE MASTER FILE ROUTINE
store "a" to soi
do while soi<>"1".and.soi<>"2".and.soi<>"3"
   erase
```

```
   @ 05,5 say"...................................."
   @ 06,5 say".                                  ."
   @ 07,5 say". SELECT ...                       ."
   @ 08,5 say".      1 ... TO USE SECTION UPDATE    ."
   @ 09,5 say".      2 ... TO USE THROUGH EOF UPDATE."
   @ 10,5 say".      3 ... TO CANCEL COMMAND        ."
   @ 11,5 say".                                  ."
   @ 12,5 say"...................................."
   ?
   wait to soi
   ? chr(7)
enddo
if soi="3"
   return
endif
erase
*** SET INITIALIZATION *****
@ 09,11 say "***********************"
@ 10,11 say "* UPDATE MASTER FILES *"
@ 11,11 say "***********************"
* SET THE FIRST RECORD TO WORK ON DAILY FILES
use pointer
store pop to popy
store 1 to count
store "y" to cond
use b:master index cody
select secondary
use b:daily
go popy
* THROUGH END OF FILE UPDATE
if soi="2".
   do while .not.eof
      store code1 to recode
      store issue to iss
      store reci-issue to add
      select primary
      find &recode
      replace onhand with onhand+add,sale with sale+iss
      if onhand<min
         replace status with "*"
      else
         replace status with " "
      endif
      select secondary
      @ 14,13 say"*****************"
      @ 15,13 say"* PLEASE WAITING *"
      @ 16,13 say"*****************"
      set talk on
      ?
      skip
      set talk off
   enddo
   store #+1 to popy
   select primary
   pack
   ? chr(7)
   use pointer
   replace pop with popy
   pack
   return
endif
* SECTION UPDATE
if soi="1"
   do while cond<>"n"
      do while count<100
         if eof
```

```
        ? chr(7)
        store #+1 to popy
        select primary
        pack
        use pointer
        replace pop with popy
        store 10000 to count
        loop
     endif
     store code1 to recode
     store issue to iss
     store reci-issue to add
     select primary
     find &recode
     replace onhand with onhand+add,sale with sale+iss
     if onhand<min
        replace status with "*"
     else
        replace status with " "
     endif
     select secondary
     @ 14,13 say "*****************"
     @ 15,13 say "* PLEASE WAITING *"
     @ 16,13 say "*****************"
     set talk on
     skip
     set talk off
     store count+1 to count
   enddo
   if count=10000
      store "n" to cond
      loop
   endif
   erase
   @ 10,4 say "TO CONTINUE UPDATE MASTER ...y/n..."
   wait to cond
   if cond="n".or.cond="N"
      ? chr(7)
      store # to popy
      select primary
      pack
      use pointer
      replace pop with popy
      loop
   endif
   store 1 to count
 enddo
 return
endif
```

```
A>type check.cmd
* CHECK DAILY FILE ROUTINE
erase
? chr(7)
@ 05,1 say" ---------------------------------------- "
@ 06,1 say".                 CAUTION                 ."
@ 08,1 say".  AFTER USE THESE SECTION YOU MUST BE LIST ."
@ 09,1 say".  DAILY FILE. IF FILED DUKE HAVE '*' MARK  ."
@ 10,1 say".  YOU MUST BE REXAMINE CODE AND NAME       ."
@ 11,1 say".                                           ."
@ 12,1 say" ---------------------------------------- "
? chr(7)
```

```
use pointer
store poi to poiy
store 1 to count
store "y" to cond
use b:master index cody
select secondary
use b:daily
go poiy
if soi="1"
   do while cond<>"n"
      do while count<100
         store code1 to check
         store name1 to rename
         select primary
         find &check
         if rename<>$(name,1,10)
            select secondary
            *MISMATCH FILL DUKE WITH "*"
            replace duke with "*"
         endif
         store count+1 to count
         select secondary
         if eof
            ? chr(7)
            store # to poiy
            use pointer
            replace poi with poiy
            store 10000 to count
            loop
         endif
         @ 14,13 say"------------------"
         @ 15,13 say"  PLEASE WAITING "
         @ 16,13 say"------------------"
         set talk on
         skip
         set talk off
      enddo
      if count=10000
         store "n" to cond
         loop
      endif
      erase
      @ 10,3 say"TO CONTINUE CHECK ... y/n ..."
      wait to cond
      if cond="n".or.cond="N"
         ? chr(7)
         store # to poiy
         use pointer
         replace poi with poiy
         loop
      endif
      store 1 to count
   enddo
   select secondary
   pack
   ? chr(7)
   return
endif
if soi="2"
   do while .not.eof
      store code1 to check
      store name1 to rename
      select primary
      find &check
      if rename<>$(name,1,10)
         select secondary
```

```
            replace duke with "*"
         endif
         select secondary
         @ 14,13 say"------------------------"
         @ 15,13 say"-    PLEASE WAITING    -"
         @ 16,13 say"------------------------"
         ?
         set talk on
         skip
         set talk off
      enddo
      store #+1 to poiy
      select secondary
      pack
      use pointer
      replace poi with poiy
      pack
      return
   endif
```

```
A>type report.cmd
* PRINT REPORT HERE
store "a" to soi
do while soi<>"1".and.soi<>"2".and.soi<>"3"
   erase
   @ 05,5 say"......................................."
   @ 06,5 say".                                     ."
   @ 07,5 say".  SELECT ...                         ."
   @ 08,5 say".       1 ... TO PRINT DAILY REPORT   ."
   @ 09,5 say".       2 ... TO PRINT MASTER REPORT  ."
   @ 10,5 say".       3 ... TO CANCEL COMMAND       ."
   @ 11,5 say".                                     ."
   @ 12,5 say"......................................."
   ?
   wait to soi
   ? chr(7)
enddo
if soi="1"
   use b:daily
   erase
   input "HOW MANY REPORTS DO YOU WANT " to d
   store 1 to adi
   do while adi<=d
      set format to print
      ? chr(27)+chr(65)
      ? chr(10)
      @ 09,19 say "********************"
      @ 10,19 say "*DAILY FILES REPORT*"
      @ 11,19 say "********************"
      ?
      @ 12,0 say "ITEM"
      @ 12,6 say "TRANSAC.No"
      @ 12,18 say "CODE"
      @ 12,30 say "NAME"
      @ 12,42 say "ISSUE"
      @ 12,50 say "RECIEVE"
      store 13 to line
      do while .not.eof
         @ line,0 say item using "XXXX"
         @ line,8 say trans using "XXXXX"
         @ line,18 say code1 using "XXXX"
```

```
          @ line,28 say name1 using "XXXXXXXXXX"
          @ line,41 say issue using "999999"
          @ line,49 say reci using "999999"
          store line+1 to line
          skip
       enddo
       go top
       store adi+1 to adi
    enddo
    @ line+1,9say"   "
    set format to screen
    return
endif
 f soi="2"
    use b:master
    do while .not.eof
       replace sale with sale/date
       skip
    enddo
    pack
    use
    use b:master
    store "a" to done
    do while done="a"
       erase
       input "HOW MANY REPORTS DO YOU WANT" to d
       store 12 to line
       store 1 to adi
       do while adi<=d
          set format to print
          ? chr(27)+chr(65)
          ? chr(10)
          @ 09,21 say"**********************"
          @ 10,21 say"*MASTER FILES REPORT*"
          @ 11,21 say"**********************"
          ?
          @ 12,0 say "CODE"
          @ 12,15 say "NAME"
          @ 12,33 say"COST"
          @ 12,41 say"ONHAND"
          @ 12,48 say"STATUS"
          @ 12,55 say"MINIMUM"
          @ 12,64 say "SALERATE"
          do while .not.eof
             @ line,0 say code using "XXXX"
             @ line,5 say name using "XXXXXXXXXXXXXXXXXXXXXXXXX"
             @ line,30 say cost using "9999999.99"
             @ line,41 say onhand using "9999999"
             @ line,50 say status using "X"
             @ line,56 say min using "999999"
             @ line,63 say sale using "9999999.99"
             store line+1 to line
             skip
          enddo
          go top
          store adi+1 to adi
       enddo
       @ line+1,8 say"   "
       set format to screen
       erase
       @ 04,3 say"..........................................."
       @ 05,3 say"                              CAUTION"
       @ 07,3 say "SOME MORE REPORT ... y/n ..."
       @ 08,3 say "IF YOU ANSWER n(NO) I WILL UPDATE MASTER"
       @ 09,3 say "SOME FIELD MAY CHANGE"
       @ 11,3 say"..........................................."
```

```
* IF ANSWER NO SALERATE AND DATE WILL CHANGE
?
wait to done
? chr(7)
if done="n"
   use b:master
   replace all sale with 0,date with 1
   pack
else
   store "a" to done
   loop
endif
enddo
return
endif
if soi="3"
   return
endif
```

ตัวอย่างต่าง ๆ ที่ยกมานี้เป็นตัวอย่างที่ได้รับการออกแบบขึ้นเพื่อใช้เป็นแนวทางในการออกแบบระบบจัดการฐานข้อมูลตามความต้องการของแต่ละบุคคลซึ่งแตกต่างกัน เช่น ระบบฐานข้อมูลการลงทะเบียนนักเรียนระบบบัญชี ระบบจ่ายเงินเดือน ระบบการจดบันทึก สรุปยอดของฝ่ายขาย ฯลฯ.ในแต่ละระบบจะมีลักษณะเฉพาะตัวอยู่มากซึ่งการแก้ไขดัดแปลงที่เป็นไปตามรูปแบบย่อมเป็นไปได้ และโดยเฉพาะอย่างยิ่งเมื่อใช้ระบบฐานข้อมูลเป็นภาษาไทยจากห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ ก็นับว่าระบบฐานข้อมูลที่ใช้จะง่ายต่อการสร้างค้นหา และใช้ประโยชน์ได้เต็มที่ ซึ่งผู้เขียนหวังว่าการใช้ไมโครคอมพิวเตอร์ในเมืองไทยจะคุ้มค่าและเกิดประโยชน์มากขึ้น

ภาคผนวก

สืบเนื่องจากโปรแกรมดีเบสทูไม่สามารถรับข้อมูลและประมวลผลข้อมูลตัวอักษรภาษาไทยได้ ทั้งนี้เพราะข้อมูลตัวอักษรภาษาไทยเป็นรหัสที่ได้สร้างขึ้นในส่วนของตารางรหัสที่เป็นส่วน 8 บิท ประจวบกับการวางรหัสตัวอักษรภาษาไทยยังไม่เป็นมาตรฐาน จึงทำให้การใช้งานในเมืองไทยดูจะไม่เป็นระเบียบนัก นับเป็นข้อยุ่งยากต่อการเลือกหาอุปกรณ์ประกอบอื่นที่ทำงานแทนกันได้ เช่นเรื่องของการใช้เครื่องพิมพ์ เป็นต้น

ห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าได้ออกแบบโปรแกรมทีเบสหรือไทย เพื่อใช้งานร่วมกับโปรแกรมดีเบสทูของบริษัทแอชตันเทต โดยส่วนของโปรแกรมไทยนี้จะเป็นส่วนการจัดการข้อมูลด้านภาษาไทยในระดับโปรแกรมจัดระบบงานซีพีเอ็มหรือเอ็มเอสดอส เช่น การจัดเรียงระดับตัวอักษร การจัดรูปแบบการพิมพ์ รวมทั้งจัดการรายละเอียดอีกหลายอย่างเกี่ยวกับโครงสร้างฐานข้อมูลที่เป็นอักษรไทยหรือไทยผสมอังกฤษ

เมื่อเรียก Thai หรือทีเบสแล้ว ถ้าในโปรแกรมนี้ไม่มีออโตโหลดเดอร์ สำหรับโหลด DBASE ผู้ใช้ต้องคีย์ DBASE เพื่อ RUN DBASE ส่วนของโปรแกรมทีเบสจึงเป็นโปรแกรมที่จะเรียกดีเบสออกมา การทำงานทุกอย่างยังคงทำให้ดีเบสทำงานได้ตามปกติ

การใช้งานโปรแกรม Thai

โปรแกรมไทยนี้จะได้รับการสร้างขึ้นและเป็นไฟล์ชนิด.COM เมื่อเริ่มใช้ในขณะอยู่ใน CP/M ต้องสอดดิสเก็ตต์ที่มีโปรแกรมไทยในช่องเสียบดิสเก็ตต์และเรียกโปรแกรมนี้ ส่วนของโปรแกรมนี้จะเข้าไปจัดการด้านข้อมูลภาษาไทย และเมื่อผู้ใช้ต้องการป้อนข้อมูลภาษาไทย (ในขณะอยู่ในโหมดภาษาอังกฤษ) ผู้ใช้จะต้องกด ESC ตามด้วย T ถ้าต้องการเปลี่ยนโหมดกลับเป็นอังกฤษกด ESC ตามด้วย E

สืบเนื่องจากการเป็นสกรีนฟอร์แม็ต หรือสกรีนดิตเตอร์ ดังนั้นจึงกำหนดให้ตัวอักษรเข้าในลักษณะเรียงกัน แต่ถ้าต้องการแสดงผลบนจอภาพเป็น 3 ระดับ ถูกอักขระ เช่นใช้ .LIST ก็จะต้องกด ESC ตามด้วย 3 แล้วจึงกด RETURN

อนึ่งหากมีการส่งข้อมูลออกทางเครื่องพิมพ์ไม่ว่ากรณีใด ถ้าหากมีตัวอักษรไทยปรากฏออกมาโปรแกรมไทยจะจัดการให้มีการพิมพ์ตามอักขระที่ถูกต้องทันที

โปรแกรมไทยนี้จะมีส่วนแยกต่างหากจากโปรแกรม DBASE ดังนั้นการทางานของ DBASE จึงยังคงปกติ โปรแกรมไทยจะเป็นส่วนของโปรแกรมจัดระบบงานและอยู่ในซีพีเอ็ม หรือ เอ็มเอสดอส

ปัญหาการใช้งานดีเบสจึงไม่น่าเกิดขึ้นเพราะโปรแกรมไทย

ส่วนในโหมดของโปรแกรมที่เขียนขึ้นใน modify command นั้น ถ้าหากต้องการส่งข้อความตัวอักษรไทยออกบนจอภาพในขณะ run โปรแกรมให้ใช้ ? CHR (2) ก่อน แล้วค่อยส่งตัวอักษรภาษาไทย และเมื่อจะกลับโหมดอังกฤษให้ใช้ ? CHR (1) ในทำนองเดียวกันถ้าคำสั่งนี้ได้รับการเอ็กโค (Echo) ออกเครื่องพิมพ์ ก็จะทำให้เครื่องพิมพ์จะพิมพ์จัดระยะห่างบรรทัดเป็นแบบไทยหรืออังกฤษได้

ตัวอย่างการใช้งานดีเบสทูในลักษณะข้อมูลภาษาไทยที่มีโปรแกรมทีเบสจัดการ

```
A>TYPE TESTB:TEST.CMD
set talk off
clear
erase
@ 1,5 SAY"----------------------------------------------------"
@ 5,5 SAY"----------------------------------------------------"
@ 9,5 SAY"----------------------------------------------------"
@ 14,5 SAY"---------------------------------------------------"
? CHR(2)
@ 3,25 SAY"<<<<< ปทานุกรมนักเรียน >>>>>"
@ 7,5 SAY"คำศัพท์          :"
? CHR(13)
@ 11,5 say"ความหมาย :"
@ 17,5 SAY"กรุณาพิมพ์คำศัพท์"
use vedio index vedy
store "y" to new
do while new="y"
       ? CHR(1)
@ 19,0 SAY"
@ 18,30 SAY" "
ACCEPT TO A
@ 6,20 SAY"
@ 7,20 SAY"
@ 8,20 SAY"
@ 10,20 SAY"
@ 11,20 SAY"
@ 12,20 SAY"
IF A=CHR(32)
   RETURN
ENDIF
```

```
    FIND &A
    ? CHR(2)
    @ 7,20 SAY VER
    @ 11,20 SAY MEAN
    ? CHR(1)
ENDDO
```

หมายเหตุ

ตัวอย่างโปรแกรมนี้สร้างไฟล์เป็นภาษาไทย ใช้ชื่อไฟล์ว่า Vedio. dbf โดยไฟล์นี้ประกอบด้วยฟิลด์เพียงสองฟิล์ด ฟิลด์แรกใช้ชื่อว่า ฟิลด์ VER หมายถึง คำศัพท์ ฟิลด์ที่สองคือ MEAN หมายถึง ความหมาย โดยโครงสร้างของไฟล์ เป็น

VER	MEAN
เสโท	เหงื่อ
อาชา	ม้า
กระจอนหู	ตุ้มหู

ก่อนการใช้โปรแกรม เราจะสร้าง index file จาก Vedio โดยใช้คำสั่ง

INDEX ON VER TO VEDY

ซึ่งก็หมายถึง เรามีไฟล์ชื่อ VEDY. NDX อยู่หนึ่งไฟล์

จากโปรแกรมบรรทัดที่ 8 คือ ? CHR (2) คือการบอกให้เครื่องรับรู้ว่าเราจะพิมพ์ข้อความภาษาไทยออกทางจอในลักษณะถูกอักขระ ซึ่งก็คือ ข้อความ "ปทานุกรมนักเรียน" ออกในบรรทัดที่ 3 คอลัมน์ที่ 25 เป็นต้นไป

การใช้คำสั่ง ? CHR (13) เป็นคำสั่งในการเคลียร์บัฟเฟอร์ของจอภาพ เพราะโดยปกติบนจอภาพของดีเบสทูจะมีบัฟเฟอร์อยู่ 80 ตัวอักษร แต่เมื่ออยู่ในโหมดตัวอักษรอาจมีข้อผิดพลาดในการนับตัวอักษรทาง physical กับทาง logical ไม่ตรงกัน การใส่การพิมพ์ CHR (13) คือ return ออกมาเพื่อเสมือนการขึ้นบรรทัดใหม่ของบัฟเฟอร์เพื่อให้บัฟเฟอร์มีที่อยู่ 80 ตัวครบ

คำสั่ง ? CHR (1) คือการกลับเข้าสู่โหมดปกติ 1 บรรทัดเหมือนภาษาอังกฤษ ส่วนการสั่งพิมพ์ข้อความว่างเปล่าในบรรทัดที่ 6, 7, 8, 10, 11, 12 คือ การเคลียร์ตัวอักษรบนจอก่อนการนำเอาข้อความใหม่มาพิมพ์ หากไม่สั่งการใน 6 บรรทัดนี้ ข้อความที่พิมพ์ทับจะทับได้ แต่ตัวที่ไม่ถูกพิมพ์ทับจะยังอยู่ ที่ต้องพิมพ์ทับ 3 บรรทัดเพราะภาษาไทยมีการพิมพ์ออก 3 บรรทัด เท่ากับ 1 บรรทัดทางการมองเห็นของดีเบส

อนึ่งการสร้างข้อความในการบรรจุลงในฟิลด์ภาษาไทย ต้องให้ขนาดของฟิลด์ภาษาไทยมากกว่าตัวอักษรที่เก็บไว้ในฟิลด์อย่างน้อย 3 ตัวอักษร หรือกล่าวอีกนัยหนึ่งคือ ต้องเว้นว่างไว้อย่างน้อย 3 ตัวอักษรตอนหลังของฟิลด์

คุณสมบัติของเครื่องไมโครคอมพิวเตอร์ที่จะใช้โปรแกรมดีเบสทู

ดีเบสทู เป็นโปรแกรมที่จะสามารถนำมาใช้กับระบบคอมพิวเตอร์ที่มีคุณสมบัติทั้งทางฮาร์ดแวร์และซอฟท์แวร์ ดังต่อไปนี้

- จะต้องเป็นระบบที่ใช้ซีพียู (CPU = Central Processing Unit) เบอร์ 8080, 8085 หรือ Z80 (เช่น จะใช้กับเครื่องยี่ห้อ NEC PC 8000, แอปเปิ้ล II ที่มี Z80 การ์ด, TRS 80/II, ออสบอนด์ 1 ฯลฯ) รวมทั้งเครื่องที่มีซีพียู 8088 เช่น เครื่องยี่ห้อ SIRIUS, IBM PC และ VICTOR 9000 เป็นต้น
- ต้องการหน่วยความจำภายในเครื่องอย่างน้อย 48 กิโลไบต์ สำหรับในเครื่องไมโครฯ โดยทั่ว ๆ ไป เพราะตัวโปรแกรมดีเบส 2 จะใช้เนื้อที่ในหน่วยความจำในตำแหน่งที่ 5CH ถึง A400H (ตัวเลขฐาน 16) แต่สำหรับบางเครื่อง เช่น แอปเปิ้ล II ต้องการหน่วยความจำอย่างน้อย 56 กิโลไบต์
- ต้องมีโปรแกรมจัดระบบงาน CP/M (version 1.4 หรือ 2.x)
- ต้องเป็นระบบที่มีดิสค์อย่างน้อยที่สุด 1 ช่อง
- คีย์บอร์ดสามารถที่จะส่งรหัสสำหรับเคลื่อนย้ายตำแหน่งของเคอร์เซอร์ ไปบนจอภาพได้ เมื่อมีการใช้ฟังก์ชัน full screen
- มีเครื่องพิมพ์เพื่อการพิมพ์ผลลัพธ์ของโปรแกรม (จะไม่มีก็ได้)

คุณสมบัติโดยทั่วไปของดีเบสทู

จำนวนเรคอร์ดต่อหนึ่งไฟล์ข้อมูล มีได้สูงสุด	65535
จำนวนตัวอักษรในหนึ่งเรคอร์ด มีได้สูงสุด	1000
จำนวนฟิลด์ในหนึ่งเรคอร์ด มีได้สูงสุด	32
จำนวนตัวอักษรในหนึ่งฟิลด์มีได้สูงสุด	254
ตัวเลขจะมีค่าสูงสุด เท่ากับประมาณ	±1.8 × 10 <63>
ตัวเลขจะมีค่าต่ำสุด เท่ากับประมาณ	±1 × 10 <-63>
การคำนวณจะมีความถูกต้องของตัวเลข	10 ตำแหน่ง
ความยาวของข้อความที่ประกอบด้วยตัวอักขระมีได้ไม่เกิน	254
คำสั่งในแต่ละบรรทัดประกอบด้วยตัวอักขระได้สูงสุด	254
ความยาวของหัวกระดาษเนื่องจากคำสั่ง REPORT มีได้สูงสุด	254
ความยาวของตัวดัชนีแบบ index hey มีได้สูงสุด	100
จำนวนของนิพจน์สูงสุดในคำสั่ง SUM	5

ชนิดของข้อมูลในระบบฐานข้อมูล

ข้อมูลที่เก็บอยู่ภายในแต่ละฟิลด์นั้น จำเป็นจะต้องกำหนดชนิดของข้อมูลไว้ให้แน่นอนในขณะที่เราใช้คำสั่งกำหนดโครงสร้างของไฟล์ (เมื่อใช้คำสั่ง CREATE) เราจะต้องกำหนดชนิดของ *ข้อมูล* เป็นอย่างใดอย่างหนึ่งใน 3 แบบ ต่อไปนี้

ชนิดตัวอักษร (Character) – ประกอบด้วยตัวอักษร ตัวเลขจำนวนเต็ม เครื่องหมายและช่องว่าง ข้อมูลชนิดนี้จะเก็บเป็นลักษณะของรหัส ASCII

ชนิดตัวเลข (Numeric) – เป็นตัวเลขบวกหรือตัวเลขลบ เป็นค่าได้ตั้งแต่ 1.0 x 10 (-63) ถึง 1.8x10 (63) ความถูกต้องเมื่อนำตัวเลขนี้ไปคำนวณได้ถูกต้องใน 10 หลัก ค่าสูงสุดที่คำนวณออกมาได้ เช่น เป็นค่า 99,999,999.99 เป็นต้น

ชนิดลอจิก (Logical) – ข้อมูลชนิดนี้จะเก็บไว้เป็นตรรก กล่าวคือ มีค่าได้เพียง 2 แบบ คือ "ถูก" หรือ "ผิด" ดีเบสทูจะถือว่ารหัสตัว ''T'', ''t'' ''Y'' หรือ ''y'' นี้จะหมายถึงค่า "ถูก" (True/Yes) และจะถือว่ารหัสตัว ''F'', ''f'', ''N'' หรือ ''n'' จะหมายถึงลอจิก "ผิด" (False/No)

การตั้งชื่อฟิลด์ของฐานข้อมูล

ฟิลด์แต่ละฟิลด์ที่อยู่ภายในระบบฐานข้อมูล จำเป็นจะต้องมีชื่อของฟิลด์นั้นกำกับอยู่ เพื่อที่จะสามารถอ้างถึงได้เมื่อต้องการ ชื่อของฟิลด์นี้จะประกอบด้วยตัวอักขระได้ไม่เกิน 10 ตัว (ไม่ใช้ช่องว่าง) และจะต้องเริ่มต้นชื่อของฟิลด์ด้วยตัวอักษรเท่านั้น (ขึ้นต้นด้วยตัวเลขไม่ได้) แล้วอาจตามมาด้วยตัวอักษรหรือตัวเลข รวมทั้งความหมายจุดคู่ (:) ซึ่งต้องไม่อยู่ตอนท้ายได้ เช่น เราอาจจะตั้งชื่อฟิลด์ ดังนี้

A	ใช้ได้
A123456789	ใช้ได้
Job:Number	ใช้ได้ (จะใช้ตัวอักษรตัวเล็กหรือตัวใหญ่ก็ได้)
Reading:	ใช้ไม่ได้ เพราะมีเครื่องหมายจุดคู่ (:) อยู่ตอนท้ายชื่อ

อย่างไรก็ตาม เราจะพยายามตั้งชื่อของฟิลด์ต่าง ๆ เหล่านี้ให้ดูแล้วมีความหมาย จะทำให้สะดวกในการเรียกใช้งานได้ดี เช่น เราตั้งชื่อฟิลด์ว่า Job:Number จะใช้งานได้สะดวกกว่าการตั้งชื่อว่า No. เฉย ๆ การอ้างถึงในภายหลังจะได้ง่าย (จำง่าย) บางครั้งเมื่อเราเขียนไฟล์คำสั่งนั้นจะสังเกตเห็นอีกว่า เราจะพยายามใช้ตัวอักษรตัวใหญ่สำหรับชุดคำสั่งต่าง ๆ แต่ส่วนชื่อฟิลด์หรือตัวแปร มักจะใช้ทั้งตัวอักษรตัวเล็กและตัวใหญ่ เพื่อจะช่วยทำให้สังเกตและแยกแยะโปรแกรมได้ชัดเจนยิ่งขึ้นนั่นเอง

ชนิดของไฟล์ที่สร้างขึ้นโดยดีเบสทู

ชื่อไฟล์ต่างที่เราสร้างขึ้นนั้น จะมีชื่อไฟล์กำกับอยู่ โดยที่ชื่อไฟล์นี้จะประกอบด้วยตัวอักขระจำนวนไม่เกิน 8 ตัว เราเรียกว่าเป็น *ชื่อไฟล์ (filename)* และความจริงแล้วยังประกอบด้วย *นามสกุลของไฟล์ (filetype* หรือบางครั้งเรียกว่า *extension)* ซึ่งจะอยู่กำกับหลังชื่อไฟล์ด้วยเครื่องหมายจุด และตามด้วยตัวอักขระอีก 3 ตัว เพื่อที่จะเป็นตัวกำหนด *ชนิดของไฟล์* นั้น

ดีเบสทูเป็นโปรแกรมที่จะช่วยทำให้ผู้ใช้ ใช้ไฟล์ต่าง ๆ เหล่านี้ได้สะดวกขึ้น แต่ไฟล์ต่าง ๆ เหล่านี้ ได้รวบรวมข้อมูลต่าง ๆ ไว้ตามจุดประสงค์ที่แตกต่างกัน ดังนั้นเมื่อสังเกตการใช้งานดีเบสทูโดยการ

ใช้คำสั่งต่าง ๆ ที่เกี่ยวข้องกับการสร้างไฟล์ นั้น เช่น คำสั่ง CREATE หรือคำสั่ง MODIFY COMMAND เป็นต้น ดีเบสทูจะถามผู้ใช้ว่าต้องการสร้างไฟล์ชื่ออะไร ผู้ใช้ก็เพียงแต่ป้อนชื่อไฟล์เท่านั้น (ไม่ต้องใส่นามสกุล) หลังจากนั้นดีเบสทูจะสร้างไฟล์นั้นขึ้นมาทันที และพร้อมกับใส่ชนิดของไฟล์หรือนามสกุลของไฟล์นั้นเองโดยอัตโนมัติ แตกต่างกันไปตามแต่คำสั่งนั้นว่าจะสร้างชนิดของไฟล์เพื่อการเก็บข้อมูลแบบไหน ดีเบสทูจะสร้างชนิดของไฟล์ได้ทั้งหมด 3 ชนิด ดังนี้

ชนิดของไฟล์	ความหมาย
.DBF ไฟล์ฐานข้อมูล	– เมื่อผู้ใช้ใช้คำสั่ง CREATE แล้วกำหนดชื่อของไฟล์พร้อมกับป้อนข้อมูลเก็บลงในไฟล์นี้นั้น ดีเบสทูจะเก็บรวบรวมข้อมูลเหล่านี้ไว้ในไฟล์ที่มีชื่อที่ กำหนด และมีนามสกุลของไฟล์เป็น .DBF โดยอัตโนมัติ ไฟล์ข้อมูล .DBF นี้ จะเก็บข้อมูลได้สูงสุด 65,535 เรคอร์ด ห้ามใช้โปรแกรมสำเร็จพวกเวิลด์โปรเซสซิ่งในการเรียกไฟล์นี้มาแก้ไข
.FRM ไฟล์รายงาน	– ไฟล์ชนิดนี้จะถูกสร้างขึ้นมาเมื่อเราใช้คำสั่ง REPORT ซึ่งจะประกอบด้วยข้อความที่เป็น *หัวกระดาษ (headings)* และข้อมูลต่าง ๆ ที่กำหนดโดยคำสั่ง REPORT นี้ เช่น TOTALS ข้อมูลในแต่ละคอลัมน์เป็นต้น ไฟล์นี้อาจจะสามารถแก้ไขได้โดยการใช้โปรแกรมเวิลด์โปรเซสซิ่ง เช่น Word Star และ ED เป็นต้น มาทำการแก้ไขข้อความภายในไฟล์ได้
.CMD ไฟล์คำสั่ง	– ไฟล์ชนิดนี้จะเป็นที่เก็บตัวโปรแกรมซึ่งเป็นชุดของคำสั่งของดีเบสทู เมื่อเราใช้คำสั่งว่า MODIFY COMMAND แล้วผู้ใช้กำหนดชื่อไฟล์ ดีเบสทูจะสร้างชื่อไฟล์ที่มีนามสกุล .CMD แล้วพร้อมที่จะรับชุดคำสั่งต่าง ๆ เข้าไปเก็บไว้ในไฟล์ ๆ นี้ ไฟล์ชนิดนี้สามารถจะสร้างขึ้นหรือถูกเรียกมาแก้ไขได้โดยใช้โปรแกรมเวิลด์โปรเซสซิ่ง หรือโปรแกรมพวกเอดดิเตอร์ได้ (Editor)
.NDX ไฟด์ดัชนี <index file>	– จะถูกสร้างโดยคำสั่ง INDEX ไฟล์ที่ถูกสร้างขึ้นนี้จะเปรียบเสมือนเป็นดัชนีที่ช่วยทำให้การทำงานของดีเบสทู เช่น การค้นหาข้อมูลตามเงื่อนไขต่าง ๆ ได้รวดเร็วมากขึ้นในขณะที่ไฟล์ฐานข้อมูลมีขนาดใหญ่มาก

.MEM ไฟล์หน่วยความจำ – ไฟล์ชนิดนี้จะถูกสร้างขึ้นโดยอัตโนมัติเมื่อเราใช้คำสั่ง SAVE ซึ่งเป็นคำสั่งในการนำค่าของตัวแปรหน่วยความจำ (memory variables) ค่าคงที่ และผลลัพธ์ของการคำนวณต่าง ๆ เก็บไว้ภายในไฟล์ชนิดนี้ เพื่อว่าเมื่อต้องการนำสิ่งเหล่านี้มาใช้อีกจึงจะใช้คำสั่งว่า RESTORE ซึ่งเป็นคำสั่งให้อ่านข้อมูลจากดิสค์ของไฟล์นี้กลับมายังหน่วยความจำของเครื่อง ไฟล์ ๆ นี้จะเก็บข้อมูลได้ถึง 64 ชุดเท่านั้น ข้อมูลแต่ละชุดจะประกอบด้วยตัวอักษรได้ไม่เกิน 254 ตัว

.TXT ไฟล์เอกสาร – ไฟล์ชนิดนี้จะถูกสร้างขึ้น เมื่อเราใช้คำสั่ง SET ALTERNATE เพื่อที่จะเก็บข้อมูลทุกอย่างที่ปรากฏบนจอภาพลงไปเก็บไว้ในดิสค์ เพื่อที่จะสามารถนำมาแก้ไข หรือนำมาพิมพ์ออกทางเครื่องพิมพ์ได้ในภายหลัง โดยอาจจะใช้เวิลด์โปรเซสซิ่ง เราอาจจะสร้างไฟล์ชนิดนี้ได้โดยการใช้คำสั่ง COPY...SDE ก็ได้

ชุดของคำสั่งในดีเบสทู

คำสั่งที่เกี่ยวกับการสร้างไฟล์

CREATE <ชื่อไฟล์> เป็นคำสั่งในการสร้างไฟล์ฐานข้อมูลไฟล์ใหม่ ผู้ใช้จะต้องกำหนดชื่อไฟล์ กำหนดโครงสร้างของไฟล์ ชื่อของฟิลด์ และชนิดของฟิลด์ที่จะเก็บข้อมูลนั้น

CREATE <ชื่อไฟล์ใหม่> FROM <ชื่อไฟล์เก่า> ทำการสร้างไฟล์ใหม่ โดยมีโครงสร้างของไฟล์เหมือนกับของไฟล์เก่า กล่าวคือ มีจำนวนฟิลด์ ชื่อของฟิลด์ และชนิดของฟิลด์ เหมือนกัน

COPY TO<ชื่อไฟล์>[FIELD <LIST>] [FOR <exp>[SDF] [STRUCTURE] ไฟล์ซึ่งอ่านมาโดยคำสั่ง USE นั้น จะถูกเก็บรวบรวมข้อมูลไว้ในอีกไฟล์หนึ่งที่เราตั้งชื่อขึ้นมานี้ โดยที่อาจจะดึงข้อมูลมาทั้งหมดหรือเฉพาะบางฟิลด์ที่เรากำหนด (หลังคำสั่งฟิลด์) หรือเฉพาะบางเรคอร์ดตามเงื่อนไขที่ต้องการ (กำหนดโดยคำสั่ง FOR) ไฟล์นี้จะมีฟอร์แมทเหมือนไฟล์เอกสารทั่วไปโดยใช้ SDF (จะได้ไฟล์ที่เป็นชนิด .TXT) หรือจะเก็บข้อมูลเฉพาะโครงสร้างเท่านั้น เมื่อใช้คำว่า STRUCTURE ถ้าไม่กำหนดเป็นอย่างอื่นจะถือว่าจะสร้างไฟล์แบบฐานข้อมูลคือเป็นไฟล์ชนิด .DBF ไฟล์นั้นเอง

MODIFY COMMAND <ชื่อไฟล์> ดีเบสทูจะสร้างไฟล์ที่มีนามสกุลของไฟล์เป็น .CMD และ

พร้อมที่จะให้ผู้ใช้ป้อนชุดของคำสั่งรวบรวมขึ้นเป็นโปรแกรมเก็บไว้ในดิสค์เป็น *ไฟล์คำสั่ง (Command file)*

MODIFY STRUCTURE

คำสั่งนี้จะทำการเปลี่ยนแปลงโครงสร้างของไฟล์ที่เราใช้คำสั่งจะมีการใช้คำสั่ง COPY เพื่อเก็บข้อมูลนี้ไปไว้ไฟล์ใหม่ก่อนการแก้ไขโครงสร้างของไฟล์ เช่น เพิ่มหรือลดจำนวนฟิลด์ในไฟล์โดยคำสั่ง MODIFY STRUCTURE นี้ เป็นต้น

REPORT [FORM <ชื่อไฟล์>] [<scope>] [TO PRINT] [FOR<exp>]

เป็นคำสั่งในการนำข้อมูลจากไฟล์ที่กำหนดชื่อนั้นมาทำการส่งรายงานผลมาแสดงผลบนจอภาพ หรือไปออกที่เครื่องพิมพ์เมื่อใช้คำสั่ง "TO PRINT" ผลลัพธ์ที่ได้อาจจะนำข้อมูลมาทั้งหมดไฟล์หรือเพียงบางเรคอร์ดตามเงื่อนไขที่กำหนดในคำสั่ง FOR เราจะสามารถกำหนดรูปฟอร์มของรายงานผลตามความต้องการได้

SAVE TO <ชื่อไฟล์>

จะทำการเก็บรวบรวมตัวแปรและค่าของตัวแปรที่เรียกว่า *ตัวแปรหน่วยความจำ (memory variables)* ซึ่งอยู่ภายในหน่วยความจำภายในของเครื่องไมโครคอมพิวเตอร์ขณะนั้น โดยเก็บไว้ในดิสค์มีชื่อไฟล์ตามที่กำหนด และชนิดของไฟล์เป็น .MEM ไฟล์

INDEX ON <คีย์> TO <ชื่อไฟล์>

เป็นคำสั่งที่จะสร้างไฟล์ใหม่ที่เรียกว่าไฟล์ *ดัชนี (index-file:-* ไฟล์นี้มีนามสกุลของไฟล์เป็น .NDX) ข้อมูลที่เก็บนี้จะเปรียบเสมือนเป็น *ข้อมูลดัชนี* ซึ่งทำให้ไฟล์ข้อมูลที่เราใช้คำสั่ง USE มานั้น สามารถถูกเรียงอันดับตามคีย์ที่กำหนดได้

JOIN TO <ชื่อไฟล์> ON <นิพจน์> [FIELDS<list>]

เป็นคำสั่งของดีเบสทูคำสั่งหนึ่งซึ่งมีประสิทธิภาพมาก โดยเป็นคำสั่งที่จะทำการนำข้อมูลจากไฟล์ 2 ไฟล์ ที่เป็นไฟล์ปฐมภูมิ และไฟล์ทุติยภูมินำมารวมกันแล้วสร้างเป็นไฟล์ที่ 3 ตามที่กำหนดชื่อไฟล์ที่ถูกสร้างขึ้นจากไฟล์ 2 ไฟล์ ที่อยู่ภายในหน่วยความจำภายในขณะนั้น จะถูกดึงออกมาสร้างเป็นไฟล์ที่ 3 เฉพาะเรคอร์ดซึ่งมีเงื่อนไขตามกำหนดเงื่อนไขหลังคำสั่ง ON และยังกำหนดฟิลด์ในแต่ละเรคอร์ดนั้นด้วย (กำหนดโดยคำสั่ง FIELSD)

TOTAL TO <ชื่อไฟล์> ON <คีย์> [FIELDS<list>]

คำสั่งนี้จะนำข้อมูลจากไฟล์ที่เรากำหนดจากคำสั่ง USE มารวบรวมเก็บไว้เป็นไฟล์ใหม่ โดยดึงมาเฉพาะฟิลด์ที่กำหนดที่สำคัญก็คือเรคอร์ดที่มีค่าของฟิลด์ที่กำหนดโดยคีย์ (หลังคำสั่ง ON) ที่เท่ากันนั้นฟิลด์อื่นในเรคอร์ดเหล่านั้นซึ่งเก็บข้อมูลชนิดตัวเลขจะถูกนำมารวมกัน แล้วเก็บไว้ในไฟล์ที่กำหนดชื่อนั้น (ไฟล์คำสั่งที่ใช้คำสั่ง USE นั้น จำเป็นจะต้องผ่านการเรียงลำดับมาก่อนด้วยคำสั่ง SORT หรือ INDEX)

ชุดคำสั่งในการเพิ่มหรือแก้ไขข้อมูล

APPEND

เป็นคำสั่งในการเพิ่มข้อมูลต่อท้ายจากเรคอร์ดสุดท้ายที่เรียกมาด้วยคำสั่ง USE พร้อมที่จะรับข้อมูลที่ผู้ใช้จะป้อนเข้าทางคีย์บอร์ด

APPEND BLANK

ทำการสร้างข้อมูลต่อท้ายเรคอร์ดสุดท้ายของไฟล์ที่ USE มาอีก 1 เรคอร์ดโดยที่เรคอร์ดที่เพิ่มขึ้นเก็บข้อมูลที่เป็น *ช่องว่าง (blank)*

APPEND FROM <ชื่อไฟล์> [FOR <นิพจน์>]

ข้อมูลจะถูกนำมาจากชื่อไฟล์ที่กำหนด มาเก็บไว้โดยต่อท้ายข้อมูลที่อยู่ในไฟล์ที่เราใช้คำสั่ง USE เราอาจจะกำหนดเงื่อนไขของการทำงานนี้ได้โดยใช้คำสั่ง FOR ต่อท้าย

BROWSE

คำสั่งนี้จะทำการอ่านข้อมูลจากไฟล์มาแสดงผลบนจอภาพ ผู้ใช้จะสามารถแก้ไขข้อมูลในแต่ละเรคอร์ดได้ทันที โดยเลื่อนตำแหน่งของเคอร์เซอร์ ไปยังจุดที่จะแก้ไขได้ทันที คำสั่งนี้จึงเป็นคำสั่งที่ช่วยในการแก้ไขข้อมูลที่เรคอร์ดไหนก็ได้ในไฟล์อย่างมีประสิทธิภาพ

DELETE FILE <ชื่อไฟล์>

ทำการลบไฟล์ที่มีชื่อตามที่กำหนดทิ้งออกจากดิสค์

DELETE [<scope>] [FOR <นิพจน์>]

ทำการลบข้อมูลตามเงื่อนไขที่กำหนดหลังคำสั่ง FOR โดยการลบข้อมูลที่กำหนดด้วย ''scope'' เช่น DELETE RECORD 2 เป็นคำสั่งที่จะลบเรคอร์ดที่ 2 (การใช้คำสั่งนี้ไม่ได้มีการลบข้อมูลทิ้งทันที เพียงแต่มีเครื่องหมายบอกการลบเท่านั้น)

EDIT <ตัวเลขบอกเรคอร์ด>

จะช่วยทำให้ผู้ใช้สามารถแก้ไขข้อมูลในเรคอร์ดที่ข้อมูลชุดที่ถูกกำหนดด้วยเครื่องหมายบอกการลบทิ้ง จะถูกลบทิ้งออกจากไฟล์จริง ๆ แล้วข้อมูลที่เหลือจะถูกจัด

วางตำแหน่งใหม่ ข้อมูลที่ถูกลบทิ้งไปแล้วนี้ จะเรียกกลับมาใหม่ไม่ได้

RECALL [<scope>] [FOR <นิพจน์>]
คำสั่งนี้จะทำงานตรงกันข้ามกับคำสั่ง DELETE กล่าวคือ ข้อมูลที่กำหนดว่าถูกลบโดยเครื่องหมายบอกการลบจะถูกยกเลิกทิ้งไปดังนั้นข้อมูลที่ถูกลบออกด้วยคำสั่ง DELETE จึงถูกเรียกกลับมาใหม่ได้ด้วยคำสั่ง RECALL นี้

RENAME <ชื่อไฟล์เดิม> TO <ชื่อไฟล์ใหม่>
ทำการเปลี่ยนชื่อไฟล์จากชื่อเดิม ให้เป็นชื่อใหม่ตามที่กำหนด

REPLACE [<scope>]<field> WITH นิพจน์, field FOR นิพจน์
เป็นคำสั่งในการนำข้อมูลซึ่งอาจจะมาจากการคำนวณของนิพจน์หรือตัวแปร หรือข้อความที่อยู่หลังคำสั่ง WITH นำไปเก็บไว้ในฟิลด์ที่กำหนดหลังคำสั่ง REPLACE เราอาจจะกำหนดเงื่อนไขการแทนค่านี้ได้โดยใช้คำสั่ง FOR ด้วย

READ
ทำให้มีการป้อนข้อมูลผ่านทางคีย์บอร์ดโดยผู้ใช้ในแบบ full screen editing กล่าวคือมีการทำงานพ่วงกับคำสั่ง c...GET... ตัวแปร โดยที่เราเลื่อนเคอร์เซอร์ไปป้อนหรือแก้ไขข้อมูลในตำแหน่งที่กำหนดได้

UPDATE FROM <ชื่อไฟล์> ON <คีย์> [ADD<field list>] [REPLACE<field list>]
เป็นคำสั่งในการแก้ไขข้อมูลภายในไฟล์ที่กำหนดโดยคำสั่ง USE มาก่อน โดยนำข้อมูลหรือบวกค่าของข้อมูลในฟิลด์ที่กำหนดก็ได้โดยใช้ ADD หรือ REPLACE ตามลำดับ การทำงานนี้จะกระทำต่อเป็นเงื่อนไขหรือคีย์ที่กำหนดแล้วคำสั่ง ON นั้นตรงกันทั้ง 2 ไฟล์

ชุดคำสั่งเกี่ยวกับการอินพุต

WAIT
หยุดการทำงาน หยุดการเลื่อนภาพบนจอ (scrolling) จะทำงานต่อไป *เมื่อกดคีย์อะไรก็ได้*

WAIT TO<ตัวแปร>
รอรับข้อมูลจากคีย์บอร์ดเพียง 1 ตัว เข้ามาเก็บในตัวแปร

INPUT "ข้อความ" TO ตัวแปร
รอรับข้อมูลทุกชนิดจากผู้ใช้ผ่านทางคีย์บอร์ด ถ้าเป็นข้อมูลชนิดตัวอักษรจะต้องอยู่ภายในป้อนอินพุตโดยคีย์บอร์ดนี้ แต่ต้องอยู่ภายในเครื่องหมายคำพูด "ข้อความ" ที่อยู่ในคำสั่งนี้เป็นเพียงสิ่งที่ส่งออกไปบนจอภาพตามข้อความนั้นก่อนรอรับอินพุต

ACCEPT "ข้อความ" TO ตัวแปร — ทำงานเหมือนคำสั่ง INPUT แต่รับข้อมูลเฉพาะชนิด ตัวอักษรเท่านั้น ผู้ใช้ไม่จำต้องใส่ข้อมูลไว้ในเครื่องหมายคำพูด

@ coord SAY "ข้อความ" GET ตัวแปร PICTURE — รับข้อมูลตามคีย์บอร์ดมาเก็บในตัวแปร ได้โดยสามารถกำหนดตำแหน่งของการอินพุตนี้บนจอภาพได้ ยิ่งกว่านั้นยังกำหนดรูปแบบหรือ "ฟอร์แมต" ของการรับข้อมูลได้โดยใช้ ''PICTUR''

ชุดคำสั่งเกี่ยวกับการเอาท์พุต

? <นิพจน์ /<"ข้อความในเครื่องหมายคำพูด"> / <ตัวแปร>

เป็นคำสั่งในการแสดงผลค่าของนิพจน์ ตัวแปร หรือข้อความภายในเครื่องหมายคำพูด ออกมาแสดงบนจอภาพ

DISPLAY [<scope>] [FOR <นิพจน์>] [OFF] — จะแสดงผลข้อมูลเฉพาะเรคอร์ดที่ต้องการตามเงื่อนไขหลังคำสั่ง FOR และถ้าเราใช้คำสั่ง OFF ตามท้ายจะหมายถึงการแสดงผลบนจอภาพ โดยไม่แสดงหมายเลขของเรคอร์ดนั้น

DISPLAY STRUCTURE — จะแสดงโครงสร้างของไฟล์ที่เราใช้คำสั่ง USE อ่านมา

DISPLAY MEMORY — แสดงผลตัวแปรหน่วยความจำ และค่าของตัวแปรในขณะที่กำลังใช้งานนั้น

DISPLAY FILES [ON <disk drive>] [LIKE<skeleton>]

แสดงผลชื่อและชนิดของไฟล์ที่มีอยู่ในช่องดิสค์ (disk drive) ที่กำหนด (เป็น A หรือ B เป็นต้น) การแสดงผลนี้จะแสดงเฉพาะไฟล์ที่มีนามสกุลเป็น .DBF เมื่อเราไม่ใช้คำสั่ง LIKE แต่ถ้าใช้คำสั่ง LIKE นั้นก็จะแสดงไฟล์ที่มีลักษณะตามโครงสร้างของ ''skeleton'' เช่น DISPLAY FILES ON B LIKE *.COM จะเป็นคำสั่งในการแสดงผลที่เป็นชื่อไฟล์ของดิสค์ช่อง B เฉพาะไฟล์ที่เป็นชนิด .COM เท่านั้น (ชื่อไฟล์อะไรก็ได้ ขอให้มี) ชนิดเป็น .COM)

@ coordinates SAY<นิพจน์/ข้อความ> GET<ตัวแปร> [USING<format>] [PICTURE<format>]

เป็นคำสั่งในการแสดงผลค่าของนิพจน์ ข้อความ หรือตัวแปร ไปแสดงที่จอภาพหรือเครื่องพิมพ์ (ขึ้นอยู่กับว่าใช้คำสั่ง SET FORMAT TO SCREEN หรือคำสั่ง SET

FORMAT TO PRINT ตามลำดับ) เราสามารถกำหนดรูปฟอร์ม (format) ได้อีกโดยใช้คำสั่ง PICTURE หรือ USING สำหรับการแสดงบนจอภาพ หรือเครื่องพิมพ์ตามลำดับ

คำสั่งในการค้นหาข้อมูล

SKIP [± <นิพจน์>] — เป็นคำสั่งในการเคลื่อนที่ไปในไฟล์ฐานข้อมูลจากเรคอร์ดเดิมที่ดีเบสทู อ่านลงมาจากดิสค์เข้าสู่หน่วยความจำภายในทีละเรคอร์ดนั้น จะมีการเลื่อนการอ่านข้อมูลจากเรคอร์ดดังกล่าวนั้นโดยเลื่อนขึ้นหรือเลื่อนลง ขึ้นอยู่กับตัวเลขของนิพจน์ที่กำหนดหลังคำสั่ง SKIP นี้

GO [TO] <ตัวเลขบอกหมายเลขเรคอร์ด> — ดีเบสทูจะทำการอ่านข้อมูลจากเรคอร์ดที่กำหนดหมายเลข

GOTO TOP — ทำการอ่านข้อมูลมาจากเรคอร์ดแรกของไฟล์ฐานข้อมูล

GOTO BOTTOM — ทำการอ่านข้อมูลจากเรคอร์ดแรกของไฟล์ฐานข้อมูล

FIND <ตัวแปรสตริง/"ข้อความ"> — จะเป็นคำสั่งในการค้นหาข้อมูลในไฟล์ฐานข้อมูลที่มีค่าตรงตามตัวแปรสตริง หรือข้อความที่กำหนดคำสั่ง FIND นี้จะทำงานได้รวดเร็วมาก แต่จะใช้งานได้ต่อเมื่อไฟล์ข้อมูลนั้นมีดัชนีไฟล์รวมอยู่ด้วย (index file)

LOCATE [<scope>] [FOR <นิพจน์>] — เป็นคำสั่งในการค้นหาเรคอร์ดที่มีเงื่อนไขตามที่กำหนดหลังคำสั่ง FOR โดยจะแสดงผลออกมาเป็นหมายเลขของเรคอร์ดแรกที่ค้นหาซึ่งเป็นเรคอร์ดที่มีเงื่อนไขตรงตามที่กำหนดไว้ในนิพจน์เงื่อนไขหลังคำสั่ง FOR และหลังจากนั้นถ้าเราใช้คำสั่ง CONTINUE ดีเบสทูจะทำการค้นหาข้อมูลในเรคอร์ดต่อ ๆ ไป เหมือนกับการทำคำสั่ง LOCATE เดิม ครั้งต่อไปนั่นเอง

คำสั่งที่เกี่ยวข้องกับตัวแปรหน่วยความจำ

ดีเบสทูจะยอมให้มีตัวแปรหน่วยความจำ รวมทั้งจำนวนฟิลด์ของข้อมูลในเรคอร์ดรวมกันได้ไม่เกิน 64 ตัว

LIST MEMORY หรือ DISPLAY MEMORY — จะเป็นคำสั่งในการแสดงข้อมูลของตัวแปรหน่วยความจำที่ใช้งานอยู่ในขณะนั้น

STORE<"ข้อความ" / ตัวเลข - นิพจน์> TO <ตัวแปร>

เป็นคำสั่งในการกำหนดค่าของตัวแปรให้มีค่าเป็นข้อความ (อยู่ในเครื่องหมายคำพูด) เป็นตัวเลข เป็นนิพจน์ที่ต้องมีการคำนวณก่อน หรือเป็นค่าของลอจิก (เป็น True (T) หรือ False (F)

RELEASE [<ตัวแปร>] [ALL]

ทำการยกเลิกตัวแปรหน่วยความจำที่กำหนด หรือลบทิ้งหมดทุกตัวแปร (ALL)

SAVE MEMORY TO<ชื่อไฟล์>

เป็นคำสั่งในการเก็บข้อมูลของตัวแปรหน่วยความจำทุกตัวมาเก็บไว้ในไฟล์ที่กำหนดชื่อโดยมีชนิดไฟล์เป็น .MEM

RESTORE FROM<ชื่อไฟล์>

ทำการอ่านไฟล์ที่กำหนดจากดิสค์ (ชื่อไฟล์ .MEM) ลงมาเก็บในหน่วยความจำ ค่าของตัวแปรที่อยู่ในหน่วยความจำเดิมจะถูกลบทิ้งไป จะได้ค่าใหม่ที่อยู่ในไฟล์ที่อ่านมาแทน

คำสั่งในการประมวลผลไฟล์

DO <ชื่อไฟล์>

ดีเบสทูจะทำการอ่านไฟล์ที่กำหนดชื่อไว้ ไฟล์นี้จะต้องเป็นไฟล์คำสั่งที่มีนามสกุลเป็น .CMD แล้วทำการเอกซิคิวท์ตามคำสั่งที่อยู่ในไฟล์นั้น (ทำงานตามโปรแกรมนั้น)

SELECT [PRIMARY] หรือ [SECONDARY]

โดยปกติแล้วผู้ใช้ดีเบสทูจะอ่านไฟล์ฐานข้อมูลได้ทีละเพียง ไฟล์เท่านั้นในแบบทั่วไป แต่ดีเบสทูจะทำให้ผู้ใช้สามารถติดต่อกับไฟล์ข้อมูลได้พร้อมกับ 2 ไฟล์ ได้โดยการกำหนดให้เป็นไฟล์ปฐมภูมิ (PRIMARY และไฟล์ทุติยภูมิ (SECONDARY) โดยคำสั่งนี้

SORT ON <ฟิลด์> TO <ชื่อไฟล์> [ASSCENDING] หรือ [DESCONDING]

เป็นคำสั่งที่จะจัดการข้อมูลที่อยู่ในไฟล์ที่เราใช้คำสั่ง อ่านมานำข้อมูลนั้นมาจัดการเรียงลำดับ ตามฟิลด์ที่กำหนด แล้วนำไปเก็บไว้ในไฟล์ใหม่ที่กำหนดชื่อไว้ โดยจะเรียงลำดับจากค่าน้อยไปหามาก หรือจากค่ามากไปหาน้อย (โดยใช้คำสั่ง ASCENDING หรือ DESCENDING ตามลำดับ แต่ถ้าเราไม่กำหนดเลยจะถือว่าเป็นการเรียงจากค่าของฟิลด์ที่น้อยที่สุดไปหามาก)

USE <ชื่อไฟล์>

เป็นคำสั่งในการอ่านไฟล์ที่มีชื่อตามที่กำหนด โดยที่มีชนิดของไฟล์เป็น .DBF ลงมาเก็บไว้หน่วยความจำ

ข้อมูลจะถูกอ่านมาจากเรคอร์ดแรกก่อน ไฟล์เดิมที่มีอยู่จะถูกปิดไฟล์ไป (close file เดิม) ดังนั้น ถ้าเราใช้คำสั่ง USE โดยไม่มีชื่อไฟล์ตามมาจะเป็นเพียงคำสั่งที่ทำการปิดไฟล์เดิมไปจากหน่วยความจำภายในเท่านั้น

USE<ชื่อไฟล์>INDEX<ชื่อไฟล์ดัชนี>

จะเป็นคำสั่งที่ทำให้ชื่อไฟล์ที่กำหนดนั้น มีความสัมพันธ์กับไฟล์ดัชนี (index file) ในลักษณะตัวชี้ตามเงื่อนไขที่ได้ทำการ ''index'' เอาไว้ เช่น เมื่อเราใช้คำสั่ง LIST หลังคำสั่งตัวอย่างข้างบนแล้วข้อมูลของแต่ละเรคอร์ดจะแสดงผลออกมาไม่เรียงตามหมายเลขเรคอร์ดในแบบปกติ แต่จะแสดงเรคอร์ดที่มีการเรียงลำดับตามที่ได้กำหนดไว้ในไฟล์ดัชนีแล้ว เป็นต้น

คำสั่งในไฟล์คำสั่ง (command file)

ACCEPT [<"ข้อความ">] TO <ตัวแปรหน่วยความจำ>

เป็นคำสั่งรอรับข้อมูลจากผู้ใช้ที่จะป้อนทางคีย์บอร์ดข้อมูลที่จะป้อนนี้เป็นลักษณะของตัวอักษรจะถูกเก็บลงในตัวแปรที่กำหนด ผู้ใช้ไม่ต้องคีย์เครื่องหมายคำพูดปิดหัวท้าย ส่วนข้อความที่อยู่ในเครื่องหมายคำพูดของตัวคำสั่งนี้ เราจะไม่ใส่ลงในคำสั่งก็ได้ ถ้าใส่ไว้ก็จะถูกส่งออกมาแสดงผลบนจอภาพก่อนที่จะรับ ข้อมูลจากคีย์บอร์ดเท่านั้น

CENCEL

เป็นคำสั่งที่จะยกเลิกหรือหยุดการทำงานตามคำสั่งในไฟล์คำสั่งนี้ แล้วออกมายังดีเบสทูตามปกติ ดีเบสทู

DO WHILE <นิพจน์เหตุผล> <ชุดคำสั่งอื่น ๆ> ENDDO

นิพจน์ที่อยู่หลังคำสั่ง DO WHILE มีการคำนวณออกมาแล้วได้คำตอบเป็นแบบลอจิก (เหตุผล) กล่าวคือ เป็นถูกหรือผิดเท่านั้น ถ้านิพจน์เหตุผลนั้นตอบว่าถูก (true) ดีเบสทูจะทำคำสั่งที่ต่อท้ายมาจนสุดคำสั่ง ENDDO แล้วจึงย้อนกลับขึ้นไปตรวจสอบเงื่อนไขของนิพจน์เหตุผลอีกครั้ง ถ้าคำตอบได้เป็นผิด (false) ดีเบสทูจะไม่ทำคำสั่งต่อท้าย DO WHILE แต่จะกระโดดข้ามมาทำคำสั่งหลัง ENNDO แทน

IF <นิพจน์เหตุผล><ชุดคำสั่ง 1>

เป็นคำสั่งในการตรวจสอบเงื่อนไขของนิพจน์เหตุผลก่อน ถ้าคำตอบได้ถูก (true) ดีเบสทูจะทำคำสั่งตามคำสั่งชุดที่ 1 ถ้าคำตอบของนิพจน์ได้ผิด (false) ดีเบสทูจะทำ

คำสั่งชุดที่ 2 ทั้ง 2 กรณีเมื่อทำงานเสร็จแล้ว จึงจะทำคำสั่งหลัง ENDIF ต่อไป

ELSE ชุดคำสั่ง 2 ENDIF — ตามคำสั่งชุดที่ 1 ถ้าคำตอบของนิพจน์ได้ผิด (false) ดีเบสทู จะทำคำสั่งชุดที่ 2 ทั้ง 2 กรณีเมื่อทำงานเสร็จแล้วจึงจะทำคำสั่ง ENDIF ต่อไป

INPUT <"ข้อความ"> TO <ตัวแปรหน่วยความจำ>

เป็นคำสั่งในการรับข้อมูลจากผู้ใช้ผ่านทางคีย์บอร์ดเก็บไว้ในตัวแปรที่กำหนด ข้อความจะเป็นชนิดตัวเลขก็ป้อนตัวเลขที่คีย์บอร์ดได้ทันทีหรือถ้าข้อมูลเป็นชนิดตัวอักษรจะต้องอยู่ภายในเครื่องหมายคำพูดถ้าข้อมูลเป็นชนิดลอจิกก็ป้อนทางคีย์บอร์ดได้เลย อาจจะเป็น T, t, Y, y สำหรับลอจิกถูกและ F, f, N, n สำหรับลอจิกผิดส่วนข้อความที่อยู่ในเครื่องหมายคำพูดตามหลัง INPUT จะมีหรือไม่ก็ได้ ถ้ามีก็เพียงแต่พิมพ์ออกมาบนจอภาพก่อนที่จะรอรับคีย์บอร์ดนั่นเอง

LOOP — เป็นคำสั่งที่บังคับให้ดีเบสทูกลับไปเริ่มต้นทำคำสั่งของ DO WHILE ใหม่

RETURN — ควรจะเป็นคำสั่งสุดท้ายของไฟล์คำสั่งทุก ๆ ไฟล์ เพราะเป็นคำสั่งที่จะบอกการสิ้นสุดการทำงานตามโปรแกรมในไฟล์คำสั่งแล้วกลับไปยังไฟล์ที่เรียกมาใช้งานนั้นในกรณีที่เราเรียกโดยดีเบสทู ก็จะกลับมาทำงานตามปกติได้ (ขึ้นมาแสดงเครื่องหมายจุด . รอรับคำสั่งต่อไป)

REMARK ข้อความ — เป็นคำสั่งที่จะทำให้มีการแสดงผลข้อความอะไรก็ได้ที่อยู่ตามหลังคำสั่งนี้ และจะดึงข้อความนั้นมาแสดงผลบนจอภาพได้ เมื่อมีการเอกซีคิวท์ก็ตามคำสั่งนี้ด้วย

WAIT [TO <ตัวแปร>] — รอรับข้อมูลจากคีย์บอร์ดเพียง 1 ตัว คงจะเป็นตัวเลขหรือตัวอักษรก็ได้ นำมาเก็บไว้ในตัวแปรที่กำหนด

NOTE ข้อความ หรือ * ข้อความ — เป็นคำสั่งสำหรับการใส่ข้อความซึ่งอาจจะเป็นคำอธิบายอยู่ภายในไฟล์คำสั่ง คำสั่งนี้ต่างจากคำสั่ง REMARK ที่ว่า จะไม่มีการแสดงผลย่อความออกมาเมื่อเอ็กซีคิวท์ผ่านคำสั่งนี้

คำสั่งพิเศษอื่น ๆ

RESET — เมื่อมีการเปลี่ยนแผ่นดิสค์ซึ่งอ่านจะเป็นดิสค์ที่เก็บข้อมูล

โดยทั่ว ๆ ไปจะอยู่ในช่อง B นั้น จะต้องมีการรีเซตระบบเพื่อให้ดีเบสทูทำความรู้จักกับดิสค์แผ่นใหม่นี้ก่อน ด้วยการใช้คำสั่ง RESET นี้เสมอ

EJECT — ควบคุมให้มีการเลื่อนกระดาษของเครื่องพิมพ์เพื่อขึ้นหน้าใหม่

QUIT[TO (ชื่อไฟล์หรือคำสั่งใน CP/M)] — เป็นคำสั่งในการยกเลิกการใช้งานหรือออกจากการทำงานของดีเบสทู กลับเข้ามายังการทำงานของโปรแกรมจัดระบบงาน (CP/M) ตามปกติ แต่ถ้าเรามีการใช้คำสั่ง TO แล้วตามด้วยคำสั่งใน CP/M หรือชื่อไฟล์ .COM แล้ว เมื่อออกจากดีเบสทูมาก็จะทำคำสั่งนั้นทันทีด้วย เช่น .QUIT TO ''SIR B:'', ''PIP PRN: = AL-''PIP PRN: = ALTERNAT.TXT'', ''DBASE'' เป็นต้น

ERASE — เป็นคำสั่งในการเคลียข้อความบนจอภาพทิ้ง

ความผิดพลาดที่ดีเบสทูแสดงออกมาเมื่อผู้ใช้ใช้คำสั่งผิด

BAD DECIMAL WIDTH FIELD — กำหนดตัวเลขหลังจุดทศนิยมในฟิลด์ชนิดตัวเลขผิด

BAD FILE NAME — ตั้งชื่อไฟล์ผิดหลัก

BAD NAME FIELD — ตั้งชื่อฟิลด์ผิดหลัก

BAD TYPE FIELD — กำหนดชนิดข้อมูลของฟิลด์ได้เป็น C, N หรือ L (ตัวอักษร ตัวเลขหรือลอจิกตามลำดับ)

BAD WIDTH FIELD — กำหนดความยาวของฟิลด์ผิด

CANNOT INSERT - THERE ARE NO RECORDS IN DATABASE FILE — ทำการแทรกข้อมูลในเรคอร์ดใหม่ไม่ได้ เพราะยังไม่มีข้อมูลในไฟล์ฐานข้อมูลเลย กรณีนี้ควรจะใช้คำสั่ง APPEND แทน

CANNOT OPEN FILE — ทำการเปิดไฟล์ไม่ได้

COMMAND FILE CANNOT BE FOUND — ค้นหาไฟล์คำสั่งที่กำหนดไม่ได้

DATA ITEM NOT FOUND — ค้นหาข้อมูลไม่เจอ

DATABASE IN USE IS NOT INDEX — ไฟล์ข้อมูลที่ใช้คำสั่ง USE มายังไม่ได้เชื่อมโยงกับไฟล์ดัชนี (มักจะเกิดจากคำสั่ง FIND)

DIRECTORY IS FULL — โปรแกรมจัดระบบงาน CP/M ไม่สามารถสร้างไฟล์บนดิสค์เพิ่มขึ้นได้ (จำนวนไฟล์มากเกินไป)

DISK IS FULL — ดิสค์เก็บข้อมูลเต็มแล้ว

END OF FILE FOUND UNEXPECTEDLY — ไฟล์ข้อมูลที่ USE มาไม่ได้มีฟอร์แมตที่ถูกต้อง ถ้าข้อมูล

	ทุกเรคอร์ดอยู่ครบและถูกต้อง ให้ใช้คำสั่ง PACK และใช้ INDEX อีกครั้ง
"FIELD" PHRASE NOT FOUND	ข้อความคำสั่งที่เกี่ยวกับ "ฟิลด์" ขาดหายไป
FILE ALREADY EXIST	มีไฟล์ข้อมูลนั้นอยู่แล้ว
FILE IS CURRENTLY OPEN	ไฟล์ยังถูกเปิดอยู่ เราจะสามารถปิดไฟล์ได้โดยการใช้คำสั่ง USE หรือ CLEAR
FORMAT FILE CANNOT BE OPENED	ไฟล์ที่ฟอร์แมตไม่สามารถเปิดได้
FORMAT FILE HAS NOT BEEN SET	ไฟล์ที่ฟอร์แมตไม่สามารถถูกกำหนดได้
ILLEGAL DATA TYPE	ใช้ข้อมูลผิดชนิด
ILLEGAL GOTO VALUE	ใช้ตัวเลขในคำสั่ง GOTO ผิด
ILLEGAL VARIABLE NAME	ชื่อตัวแปรและชื่อของฟิลด์ จะต้องด้วยตัวอักษรหรือตัวเลขหรือเครื่องหมายจุดคู่ (COLON :) เท่านั้น
INDEX DOES NOT MATCH DATABASE	ดีเบสไม่สามารถสอดคล้องกับคีย์ที่ทำเป็นดัชนี ให้ลองใช้ไฟล์ดัชนีไฟล์อื่นบ้าง
JOIN ATTEMPTED TO GENERATE MORE THAN 65,534 RECORDS	มีการสร้างเรคอร์ดมากกว่า 65,534 เรคอร์ด โดยการใช้คำสั่ง JOIN
KEYS ARE NOT THE SAME LENGTH	คีย์มีความยาวไม่เท่ากัน
MACRO IS NOT A CHARACTER STRING	คำสั่ง & จะต้องเป็นข้อมูลชนิดตัวอักษรเท่านั้น
MORE THAN 5 FIELDS TO SUM	มีการใช้คำสั่ง SUM ของฟิลด์มากกว่า 5 ฟิลด์
NESTING LIMIT VIOLATION EXCEEDED	
NO EXPRESSION TO SUM	ไม่มีนิพจน์ในคำสั่ง SUM
NO "FOR" PHRASE	ไม่มีคำสั่ง FOR
NO "FROM" PHRASE	ไม่มีคำสั่ง FROM
NO FIND	ดีเบสค้นหาข้อมูลที่กำหนดไม่พบ
NON-NUMERIC EXPRESSION	นิพจน์ไม่ได้เป็นตัวเลข
NONEXISTENT FILE	ไม่มีไฟล์นั้นอยู่
"ON" PHRASE NOT ROUND	ไม่มีคำสั่ง ON อยู่
OUT OF MEMORY FOR MEMORY VARIABLES	ไม่มีเนื้อที่ในหน่วยความจำสำหรับตัวแปรหน่วยความจำ
RECORD LENGTH EXCEEDS MAXIMUM SIZE (OF 1000)	จำนวนข้อมูลในเรคอร์ดมีมากเกิน 1000 ตัว
RECORD NOT IN INDEX	ไฟล์ดัชนีไม่ได้ถูกเปลี่ยนแปลงใหม่ (UPDATE) หลังจากที่มีการเพิ่มข้อมูลใหม่ลงไป ดังนั้นควรจะใช้คำสั่ง

	INDEX อีกครั้ง
RECORD OUT OF RANGE	หมายเลขของเรคอร์ดมีค่าสูงกว่า เรคอร์ดสูงสุดของไฟล์ข้อมูล
SORTER INTERNAL ERROR, NOTIFY SCDP	ความผิดพลาดภายในเนื่องจากตัวดีเบสทูเอง
SOURCE AND DESTINATION DATA TYPES ARE DIFFERENT	ไฟล์ต้นฉบับและไฟล์เป้าหมาย มีชนิดของข้อมูลไม่เหมือนกัน
SYNTAX ERROR	เขียนรูปแบบของคำสั่งผิดหลัก
SYNTAX ERROR IN FORMAT SPECIFICATION	รูปแบบของคำสั่งผิดไปจากฟอร์แมตที่กำหนด
SYNTAX CRROR, ER-ENTER	ป้อนข้อมูลผิด ให้ทำการป้อนใหม่
"TO" PHRASE NOT FOUND	คำสั่ง "TO" ไม่มี
TOO MANY CHATACTERS	มีตัวอักษรมากเกินไปในคำสั่ง
TOO MANY FILES ARE OPEN	เราจะสามารถเปิดไฟล์ได้พร้อมกันไม่เกิน 16 ไฟล์
TOO MANY MEMORY VARIABLES	เราจะใช้ตัวแปรหน่วยความจำได้สูงสุด 64 ตัวแปร
TOO MAMY RETURNS ENCOUNTERED	อาจจะเป็นความผิดพลาดเนื่องจากโครงสร้างของไฟล์คำสั่งผิดไป
"WITH" PHRASE NOT FOUND	ตกคำสั่ง "WITH" ไป
UNASSIGNED FILE NUMBER	ความผิดพลาดภายในระบบดีเบสทูเอง
*** UNKNOWN COMMAND	คำสั่งที่ดีเบสทูไม่รู้จัก
VARIABLE CANNOT BE FOUND	ตัวแปรตัวนั้นไม่มี เราควรจะตรวจสอบชื่อของตัวแปรนั้นอีกครั้ง

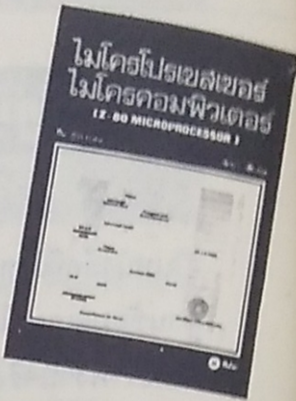
หนังสืออ้างอิง

1. Date, C.J., An Introduction to Database Systems, Addison-Wesley Publishing Company, Massachusetts, 1981.
2. Reference Manual for dBASEII, Ashton-Tate, Culver City, CA 90230, 1981.
3. User's Guide for dBASE II, Ashton-Tate, Culver City, CA 90230, 1981.

หนังสือสำหรับผู้ศึกษา/ใช้/เล่นคอมพิวเตอร์

ไมโครโปรเซสเซอร์-ไมโครคอมพิวเตอร์

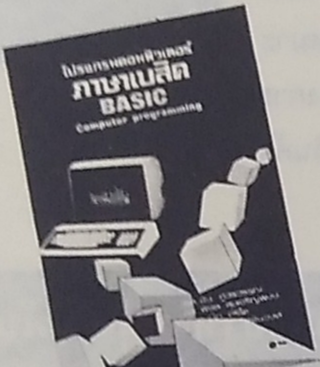
โดย ยืน ภู่วรวรรณ และวัฒนา เชียงกูล ราคา 75 บาท



หนังสือเล่มนี้เป็นการวางพื้นฐานของไมโครคอมพิวเตอร์ โดยเน้นให้เห็นถึงโครงสร้างคอมพิวเตอร์ที่จำเป็นต้องมีซีพียูหน่วยความจำ และอินพุท-เอาท์พุท นอกจากนี้ยังกล่าวถึงรายละเอียดของซีพียู Z-80 และการต่อเป็นระบบไมโครคอมพิวเตอร์ การเขียนโปรแกรม การประยุกต์ใช้งาน และสุดท้ายจะเป็นบทที่ว่าด้วยอุปกรณ์สนับสนุน ระบบไมโครคอมพิวเตอร์ที่สำคัญซึ่งได้แก่ 8214, 8212, Z-80 PIO, Z-80 CTC และ 8251

โปรแกรมคอมพิวเตอร์ภาษาเบสิค

โดยยืน ภู่วรวรรณ พิชิต สุขเจริญพงศ์ สมนึก คีรีโตและภิญโญ ศรีสุชินวงศ์ ราคา 55 บาท



เป็นหนังสือภาษาเบสิคอีกเล่มหนึ่งที่จัดทำขึ้น โดยคณะอาจารย์จากห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตร โดยพยายามสร้างพื้นฐานการเขียนโปรแกรมสำหรับผู้เริ่มต้น การประยุกต์ใช้งานของภาษาเบสิคในเรื่องการเขียนภาพ การสร้างเกมการใช้งานทางคณิตศาสตร์และสถิติ การใช้งานทางด้านธุรกิจ การบริหาร

รวมโปรแกรมสำหรับแอปเปิ้ล เล่ม 1

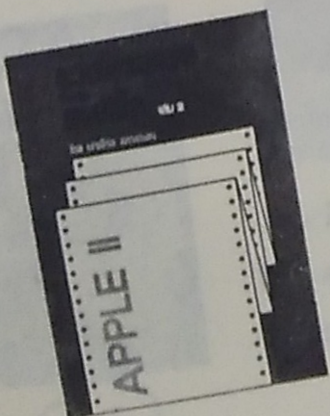
โดย เฮาเอิร์ด เบเรนบอน ราคา 48 บาท



และนี่คือ โปรแกรมใช้งานสำหรับนักเล่นไมโครคอมพิวเตอร์ที่ต้องการจะใช้เครื่อง APPLE II ให้คุ้มค่า มี 28 โปรแกรมอยู่ภายในเล่ม แบ่งเป็น 6 ภาค ตามลักษณะการใช้งาน เช่น ภาคเกี่ยวกับเวลา ธุรกิจและการลงทุน การศึกษา เป็นต้น ตัวอย่างโปรแกรม ได้แก่เครื่องควบคุมการปิดเปิดไฟฟ้าสารพัดประโยชน์ เครื่องตั้งเวลาอัตโนมัติ และคิดค่าเสื่อมราคาวิเคราะห์การใช้ไฟฟ้า หมอดูไพ่ป๊อก และเกมฝึกสมองอีกมากมาย

รวมโปรแกรมสำหรับแอปเปิ้ล เล่ม 2

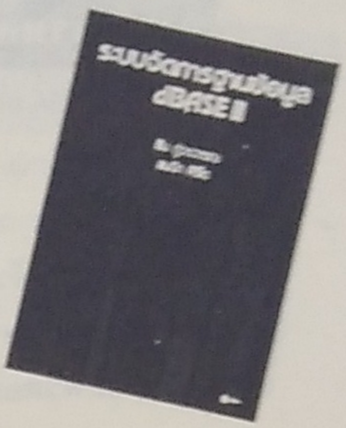
โดย เฮาเอิร์ด เบเรนบอน ราคา 55 บาท



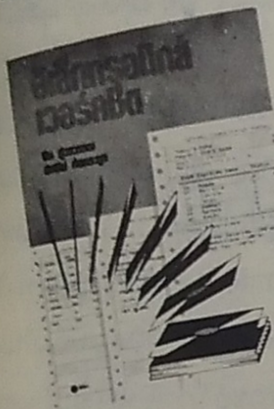
เป็นหนังสือรวมโปรแกรมใช้งานเครื่อง APPLE II อีก 37 โปรแกรม อาทิเช่น การคิดดอกเบี้ยทบต้น โปรแกรมเก็บหมายเลขโทรศัพท์ โปรแกรมรักษาสภาพสมดุลของบัญชีเช็ค โปรแกรมแนะนำการซื้อหุ้นทดสอบความสามารถทางจิต โปรแกรมแบ่งเกรดนักเรียน และโปรแกรมทดสอบความสามารถอีกมากมาย

ระบบจัดการฐานข้อมูล dBASE II

โดย ยืน ภู่วรวรรณ และสมนึก คีรีโต   ราคา 60 บาท



เป็นหนังสือสำหรับผู้ที่ยังไม่เคยใช้ไมโครคอมพิวเตอร์ในระบบจัดการฐานข้อมูลมาก่อนเลย และสนใจการใช้งาน dBASE II ซึ่งเป็นโปรแกรมที่มีประสิทธิภาพสูงสุดและได้รับความนิยมมากโปรแกรมหนึ่ง ภายในเล่มแบ่งเป็น 3 ตอน ตอนแรกจะวางพื้นฐานของระบบจัดการฐานข้อมูล ตลอดจนโปรแกรมโครงสร้าง ตอนที่สองว่าด้วยวิธีการทำงานของ dBASE II และตอนที่สามว่าด้วยการนำเอา dBASE II มาใช้งาน



อิเล็กทรอนิกส์เวอร์คชีต

โดย ยืน ภู่วรวรรณ และอัศนีย์ ก่อตระกูล   ราคา 59 บาท

เหมาะสำหรับผู้ที่เรียนรู้การใช้งานคอมพิวเตอร์มาบ้างแล้ว และกำลังต้องการนำไมโครคอมพิวเตอร์มาใช้งานจริง โดยจะให้หลักการและวิธีใช้อิเล็กทรอนิกส์เวอร์คชีต ซึ่งใช้ได้ทั้งวิสิแคลค์ มัลติแพลน ซูเปอร์แคลค์ (ทั้งกำลัง 2 และกำลัง 3) และโลตัส 123 ตัวอย่างของอิเล็กทรอนิกส์เวอร์คชีตในหนังสือเล่มนี้ใช้กับโปรแกรมซูเปอร์แคลค์เป็นหลัก แต่ก็มีหลักการใช้หรือการเรียกคล้ายอิเล็กทรอนิกส์เวอร์คชีตอีกหลายตัว

ซีพีเอ็ม (CP/M Operating System)

โดย ยืน ภู่วรวรรณ พิชิต สุขเจริญพงษ์ และสมนึก คีรีโต ราคา 58 บาท



ภายในหนังสือเล่มนี้ได้ให้รายละเอียดเกี่ยวกับโครงสร้างภายในของซีพีเอ็ม การเชื่อมต่อทางซอฟต์แวร์เพื่อพัฒนาโปรแกรมหรือดัดแปลงแก้ไขโปรแกรมการจัดการระบบข้อมูลบนแผ่นดิสเก็ตต์ รายละเอียดเกี่ยวกับคำสั่งต่าง ๆ ที่ใช้ในซีพีเอ็ม ทั้งเป็นคำสั่งที่อยู่ภายในตัวซีพีเอ็มเอง และคำสั่งประกอบที่อยู่ภายนอก ทำให้ผู้ศึกษาสามารถเรียนรู้และเข้าใจเกี่ยวกับการพัฒนา หรือเขียนโปรแกรมภาษาแอสเซมบลี้ได้เป็นอย่างดี

โปรแกรมคอมพิวเตอร์ภาษาฟอร์แทรน

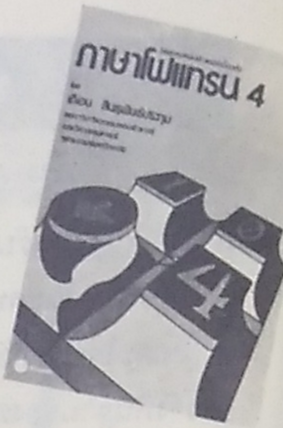


โดยยืน ภู่วรวรรณ พิชิต สุขเจริญพงษ์ และนนทวัฒน์จันทร์เจริญ ราคา 68 บาท

หนังสือเล่มนี้เหมาะสำหรับใช้เป็นตำราในระดับวิทยาลัย มหาวิทยาลัย ตลอดจนผู้ที่จะศึกษาด้วยตนเอง เนื้อหาภายในได้วางโครงร่างในลักษณะต่อเรียงกันไปโดยเริ่มจากหลักการทางคอมพิวเตอร์ สิ่งที่ควรรู้เกี่ยวกับคอมพิวเตอร์ หลักการทางด้านภาษาคอมพิวเตอร์ การเขียนแผนภูมิสายงานหรือผังงานการใช้คำสั่งต่าง ๆ ที่มีในภาษาฟอร์แทรนและตอนสุดท้ายว่าด้วยการประยุกต์ภาษาฟอร์แทรนมาใช้งาน

โปรแกรมคอมพิวเตอร์เบื้องต้น ภาษาโฟแทรน 4

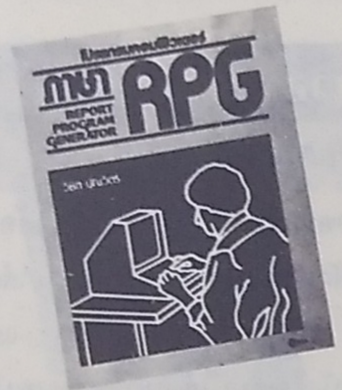
โดย เดือน สินธุพันธ์ประทุม ราคา 49 บาท



เป็นหนังสืออีกเล่มหนึ่งที่ได้รับความสนใจอย่างสูงจากผู้อ่านจนต้องพิมพ์เป็นครั้งที่ 4 เนื่องจากเนื้อหาสามารถอ่านเข้าใจได้ง่าย ทั้งผู้เริ่มเรียนหรือผู้ที่พอมีพื้นความรู้มาบ้างแล้ว คำสั่งพื้นฐานจากหนังสือเล่มนี้สามารถใช้ได้กับคอมพิวเตอร์ทุกเครื่องที่รับภาษาโฟแทรน 4 ได้ และยังเพิ่มเติมคำสั่งบางประเภทซึ่งใช้ได้เฉพาะเครื่องคอมพิวเตอร์ที่มีประสิทธิภาพสูงด้วยในบทที่ 10

โปรแกรมคอมพิวเตอร์ ภาษา RPG

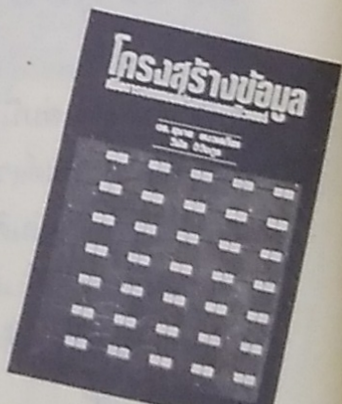
โดย วิชิต ปุณวัตร์ ราคา 140 บาท



ภาษา RPG มีลักษณะที่เหมาะสมกับงานธุรกิจเป็นส่วนใหญ่ ทั้งนี้เพราะสามารถเขียนรายงานแบบต่าง ๆ ทางธุรกิจได้ง่ายและรวดเร็วและยังมีคุณลักษณะที่ดีเด่นอีกหลายประการ ภาษา RPG ได้รับการปรับปรุงแก้ไขโดยการนำเอาเทคนิคการเขียนโปรแกรมโครงสร้าง(Structured Programming) มาประยุกต์ใช้ ทำให้ RPG สามารถเขียน Program Flow ได้ และทั้งภาษา RPG ได้ถูกปรับปรุงเป็น RGP II และ RPG III ตามลำดับ ทำให้ขีดความสามารถของภาษาสูงขึ้นเป็นอย่างมาก จนกล่าวว่าถ้ารู้จักใช้แล้วจะติดใจ

โครงสร้างข้อมูล

โดย สุชาย ธนวเสถียร และวิชัย จิวังกูร ราคา 54 บาท



หลังจากที่เรียนภาษาคอมพิวเตอร์ ตลอดจนการเขียนโปรแกรมแล้ว ขั้นต่อไปคือเรียนวิชาโครงสร้างข้อมูล วิชานี้มีความสำคัญอย่างมากเนื่องจากเป็นวิชาการพื้นฐานทางคอมพิวเตอร์ที่จะช่วยให้ผู้เขียนโปรแกรมทุกคนสามารถใช้โครงสร้างข้อมูลแบบที่เหมาะสม ซึ่งช่วยให้โปรแกรมทำงานเร็วขึ้น กินเนื้อที่น้อยลง และมีประสิทธิภาพตามที่ต้องการ ฉะนั้นถ้าท่านได้เริ่มเรียนรู้การใช้คอมพิวเตอร์โดยใช้เครื่องไมโคร ไม่ว่าจะเป็น APPLE จริงหรือเทียม หรือเครื่องยี่ห้อใดก็ตาม หนังสือเล่มนี้จะเป็นเล่มที่คุณพลาดไม่ได้



คอมพิวเตอร์กับการเขียนโปรแกรมภาษาเบสิค

โดย พรทิพย์ เลาหวิโรจน์ และสุพจน์ จิตต์ประเสริฐ ราคา 65 บาท

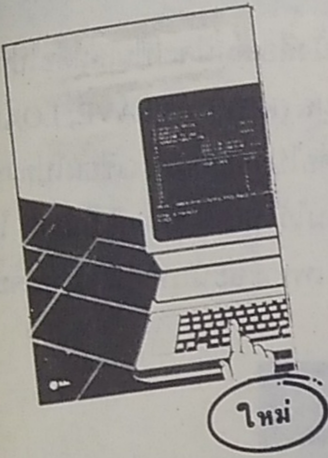
หนังสือเล่มนี้สำหรับผู้ที่ต้องการเรียนและเขียนภาษา APPLESOFT BASIC ด้วยตนเองให้เป็นภายใน 50 ชั่วโมง โดยจะเริ่มต้นตั้งแต่พื้นฐานของการโปรแกรมด้วยภาษาเบสิค อ้างอิงถึงการใช้งานกับเครื่องยี่ห้อ APPLE โดยจะค่อย ๆ พัฒนาคุณขึ้นมาจนถึงระดับที่จะโปรแกรมงานบางอย่างได้เอง

เรื่องน่ารู้เกี่ยวกับไมโครคอมพิวเตอร์

โดย ยืน ภู่วรวรรณ ราคา 53 บาท



เหมาะสำหรับผู้ที่อยากรู้รายละเอียดทั่วไปเกี่ยวกับไมโครคอมพิวเตอร์ เพื่อศึกษาข้อมูลรายละเอียดต่าง ๆ โดยเนื้อหาจะเน้นในลักษณะเรื่องทั่วไปที่เกี่ยวข้องถึงฮาร์ดแวร์ ซึ่งได้แก่ส่วนประกอบต่าง ๆ และส่วนของซอฟท์แวร์ที่ใช้กับไมโครคอมพิวเตอร์ จึงเปรียบเสมือนคู่มือที่ผู้สนใจในระดับพื้นฐานควรมีไว้เพราะสามารถศึกษารายละเอียดต่าง ๆ รวมทั้งเป็นข้อมูลสำหรับการจัดซื้อ การพิจารณาสเปค และสร้างฐานความรู้เพื่อศึกษาไมโครคอมพิวเตอร์ในระดับที่ลึกซึ้งขึ้นไป



มัลติแพลน

โดย ยืน ภู่วรวรรณและดร.พิชิต สุขเจริญพงษ์ ราคา70 บาท

เป็นหนังสือที่มีเนื้อหาเน้นสำหรับผู้ที่จะนำเอามัลติแพลนไปใช้งาน ต้องการเข้าใจมัลติแพลนเป็นอย่างดีและใช้ประโยชน์จากมัลติแพลนได้อย่างเต็มที่ รายละเอียดและเนื้อหาเริ่มจากระดับผู้เริ่มต้นจนถึงชั้นระดับผู้ใช้งานจริง เนื้อหาทั้งหมดจัดได้ว่าละเอียดทุกขั้นตอนจนกล่าวได้ว่า เหมาะที่ผู้ใช้มัลติแพลนจะมีไว้เป็นหนังสือที่ประกอบการใช้ได้เป็นอย่างดี

เวิร์ดโปรเซสซิ่ง

โดย ยืน ภู่วรวรรณ ดร.พิชิต สุขเจริญพงษ์ ราคา 65 บาท



เป็นหนังสือที่ว่าด้วยหลักการของเวิร์ดโปรเซสซิ่ง โดยได้ให้รายละเอียดสำหรับผู้ศึกษา ซึ่งเริ่มจากไม่มีพื้นฐานทางคอมพิวเตอร์เลย หรือผู้ที่รู้คอมพิวเตอร์มาบ้างแต่ต้องการใช้เวิร์ดโปรเซสซิ่งให้มีประสิทธิภาพ รายละเอียดของหนังสือเล่มนี้จึงจัดได้ว่ามีความละเอียดในเนื้อหามาก โดยได้กล่าวถึงทั้งเวิร์ดสตาร์ เมลเมิร์จและสเปลสตาร์ ซึ่งหาได้ยากในหนังสือประเภทเดียวกัน นอกจากนี้ยังได้กล่าวถึงวิธีการใช้ไทยเวิร์ดโปรเซสซิ่งที่ชื่อว่าขวัญใจนักพิมพ์ดีด



เรียน APPLESOFT BASIC ด้วยตนเอง (เล่ม 1)

โดย พ.ต.ประพัฒน์ อุทโยภาศ ราคา 45 บาท

หนังสือเล่มนี้จะช่วยให้ท่านสามารถเขียนโปรแกรมสั่งงานไมโครคอมพิวเตอร์ได้ด้วยตัวท่านเอง โดยจะเริ่มตั้งแต่คอมพิวเตอร์คืออะไร เป็นอย่างไร มีอะไรบ้าง จะต้องสั่งงานอย่างไร คำสั่งที่ใช้มีอะไรบ้าง ตลอดเล่มจะมีโปรแกรมที่ทำการทดสอบแล้ว โดยผ่านเครื่อง APPLE II ให้ท่านฝึกฝนโดยเริ่มจากง่าย ๆ ขึ้นไป จนในที่สุดท่านก็จะสามารถอ่านโปรแกรมเข้าใจได้ด้วยตัวท่านเอง

เรียน APPLESOFT BASIC ด้วยตนเอง (เล่ม 2)

โดย พ.ต.ประพัฒน์ อุทโยภาศ ราคา 42 บาท



เป็นหนังสือเพิ่มความรู้สำหรับผู้ที่อ่านเล่ม 1 มาแล้ว เพื่อจะได้ศึกษาส่วนที่ผู้เขียนเว้นไว้ เพื่อมิให้สับสนในระยะที่เริ่มเรียนรู้ บทต่าง ๆ ภายในเล่มมีลึกเข้าไปในคอมพิวเตอร์, เพิ่มเติมเสริมแต่ง, FUNCTIONS, กราฟิก, SHAPE TABLE, TEXT FILE และ FLOW CHART

เรียน/เล่น/ใช้ โปรแกรม APPLESOFT BASIC

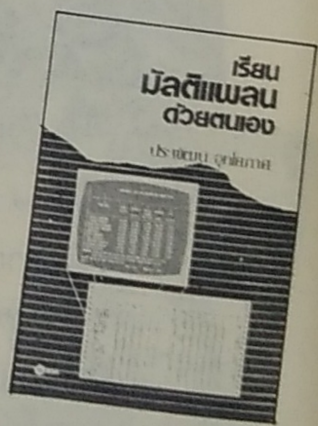
โดย พ.ต.ประพัฒน์ อุทโยภาศ ราคา 45 บาท



เมื่อท่านสามารถอ่านโปรแกรมเข้าใจแล้ว หนังสือเล่มนี้จะเป็นเล่มต่อไปที่ช่วยเพิ่มประสบการณ์แก่ท่านในด้านการหัดป้อนข้อมูล (KEY) หัด SAVE, LOAD, LIST และ RUN หัดแก้และหัดแปลงโปรแกรม ซึ่งต่อไปก็จะสามารถเขียนโปรแกรมได้เองสำหรับโปรแกรม 40 โปรแกรม ที่ได้รวบรวมไว้ในเล่ม มีทั้งที่เป็นเกม โปรแกรมทางการศึกษา คณิตศาสตร์ สถิติ และธุรกิจ เพราะฉะนั้นท่านสามารถจะเลือกฝึกได้อย่างจุใจ

เรียนมัลติแพลนด้วยตนเอง

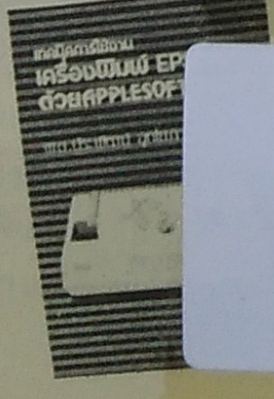
โดย ประพัฒน์ อุทโยภาศ ราคา 46 บาท



หากท่านมีนิ้วมือเหลืออย่างน้อย 2 นิ้ว พออ่านภาษาอังกฤษออก มีไมโครคอมพิวเตอร์ที่มีหน่วยความจำอย่างน้อย 64 KBYTE มีโปรแกรมสำเร็จรูปชื่อมัลติแพลน และได้อ่านหนังสือเล่มนี้เพียงครึ่งเล่ม ท่านก็สามารถนำคอมพิวเตอร์ไปใช้ประโยชน์ได้อย่างกว้างขวาง อาทิเช่น ทำบัญชี สินค้าคงคลัง ออกอินวอยซ์ เก็บหมายเลขโทรศัพท์ ฯลฯ หนังสือเล่มนี้ใช้ภาษาง่าย ๆ อย่างเป็นกันเองเหมือนเพื่อนคุยกัน แต่ขณะเดียวกันจะให้ความรู้ที่แม้แต่ในหนังสือคู่มือการใช้โปรแกรมนี้เองก็ไม่ได้กล่าวไว้ กล่าวได้ว่าหนังสือเล่มนี้จะเป็นประโยชน์แก่ทุกคน ตั้งแต่ผู้เริ่มใช้ จนกระทั่งผู้ที่ได้เคยใช้โปรแกรมนี้มาอย่างชัดเจน

เทคนิคการใช้งานเครื่องพิมพ์ EPSON ด้วย APPLESOFT BASIC

โดย พ.ต.ประพัฒน์ อุทโยภาศ ราคา 22 บาท



...คอมพิวเตอร์ตระกูล
...ระยะบรรทัด การ
...บัตรสำเร็จรูป การ
...หนังสือทั้งไทยและ
...เล่มนี้เหมาะสำหรับ

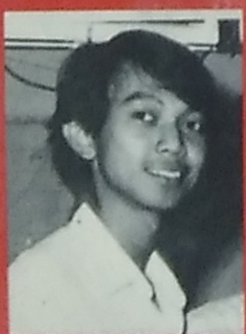
ระบบจัดการฐานข้อมูลไม่ใช่เป็นเพียงระบบการเก็บข้อมูลเข้าไฟล์ แล้วค้นจากไฟล์เท่านั้น แต่ยังรวมไปถึงการวางโครงสร้างการจัดเก็บให้มีประสิทธิภาพ ให้ค้นหาได้อย่างรวดเร็ว ประหยัดเนื้อที่เก็บ และมีการบริหารข้อมูลเพื่อสร้างผลลัพธ์ให้เป็นไปตามความต้องการของผู้ใช้ด้วย ดังนั้นระบบจัดการฐานข้อมูลจึงเป็นเรื่องที่สลับซับซ้อนอยู่มาก

หนังสือเล่มนี้เหมาะสำหรับผู้ที่ยังไม่เคยใช้ไมโครคอมพิวเตอร์ในระบบจัดการฐานข้อมูลมาก่อนเลย แต่สนใจการใช้งาน dBASE II ซึ่งเป็นโปรแกรมจัดการฐานข้อมูลที่มีประสิทธิภาพสูงสุดและได้รับความนิยมมากโปรแกรมหนึ่ง ภายในเล่มแบ่งเป็น 3 ตอน ตอนแรกจะวางพื้นฐานของระบบจัดการฐานข้อมูลตลอดจนโปรแกรมโครงสร้าง ตอนที่สองว่าด้วยวิธีการทำงานของ dBASE II และตอนที่สามว่าด้วยการนำเอา dBASE II มาใช้งาน

เกี่ยวกับผู้เขียน



ยืน ภู่วรวรรณ จบการศึกษาปริญญาตรี (เกียรตินิยม) จากจุฬาลงกรณ์มหาวิทยาลัยในสาขาวิศวกรรมไฟฟ้าสื่อสารเมื่อปี พ.ศ. 2516 จบปริญญาโทในสาขาวิศวกรรมไฟฟ้าจากจุฬาฯ และปริญญาโททางวิศวกรรมอุตสาหการรมจาก AIT ในปี พ.ศ. 2519 ปัจจุบันดำรงตำแหน่งรองศาสตราจารย์และหัวหน้าห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ ผลงานที่สำคัญ ได้แก่ การพัฒนาระบบโปรแกรม การประยุกต์ไมโครคอมพิวเตอร์เพื่อใช้งานในภาษาไทยและในระบบควบคุม หุ่นยนต์อุตสาหกรรม ฯลฯ นอกจากนี้แล้วยังมีผลงานด้านการเขียนบทความและเขียนตำราทางอิเล็กทรอนิกส์และทางคอมพิวเตอร์อีกมากมาย และยังเป็นวิทยากรให้กับหน่วยงานของรัฐบาลและเอกชนอีกมากมายเช่นกัน



สมนึก คีรีโต จบการศึกษาปริญญาตรีในสาขาวิศวกรรมไฟฟ้า ... มหาวิทยาลัยเกษตรศาสตร์เมื่อปี พ.ศ. 2523 ผ่านการดูงานด้านหุ่นยนต์ ... ญี่ปุ่น ปัจจุบันทำงานในหน้าที่นักวิจัยและอาจารย์ประจำห้องปฏิ... คอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ มีผลงานการวิจัย... ภาษาไทยบนไมโครคอมพิวเตอร์ หุ่นยนต์อุตสาหกรรม การ... คอมพิวเตอร์ ฯลฯ และยังมีผลงานการเขียนบทความและตำรา... ให้กับหน่วยงานหลายแห่งอีกด้วย

ISBN